

# COMPUTER AIDED SIMULATION OF THE DYNAMICS OF A MANIPULATOR

by

MADHU SHEKHAR, C.

ME TH1  
me / 1985 / m  
SH42C

1985

M

SHE

COM



DEPARTMENT OF MECHANICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR  
MAY, 1985

# COMPUTER AIDED SIMULATION OF THE DYNAMICS OF A MANIPULATOR

A Thesis Submitted  
In Partial Fulfilment of the Requirements  
for the Degree of  
MASTER OF TECHNOLOGY

by  
MADHU SHEKHAR, C.

to the  
DEPARTMENT OF MECHANICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR  
MAY, 1985

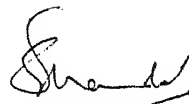
17 JUN 1983  
I.I.T. KANPUR  
CENTRAL LIBRARY  
A 87574

ME-1905-M-SHE-COM

# CERTIFICATE

Certified that the present work entitled  
'Computer Aided Simulation of the Dynamics of a  
Manipulator' has been carried out by Sri Madhu Shekhar, C.  
~~under~~ the supervision of Dr. B.Sahay, and has not been  
submitted elsewhere for a degree.

Dr. B.Sahay has gone abroad to Canada for  
three months, and has authorised me to submit the  
certificate on his behalf and to look after the viva  
of Sri Madhu Shekhar.



S.G.Dhande  
Assistant Professor  
Mechanical Engg. Department  
IIT, Kanpur

May 1985

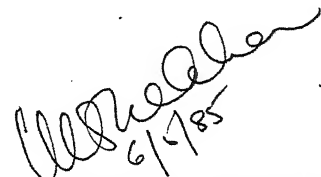


## ACKNOWLEDGEMENT

I acknowledge with deep gratitude the invaluable help and constant motivation provided to me by my Thesis Advisor Dr. B. Sahay, without which this work would not have reached completion.

I am also highly indebted to Dr. S.G.Dhande, who readily consented to conduct my oral examination in the absence of Dr. B. Sahay.

Various other people have contributed in various ways towards this project : The Computer Centre, Library, and Mechanical Department staff, and of course my friends.

A handwritten signature in dark ink, appearing to read 'C. Madhu Shekhar', with the date '6/7/85' written below it.

C.MADHU SHEKHAR

## CONTENTS

	Page
List of symbols	i
Abstract	ii
1. INTRODUCTION	1
1.1 Introduction	1
1.2 Literature Review	3
1.3 Outline of the present work	6
2. MODELLING	10
2.1 Introduction	10
2.2 Geometric Model	13
2.3 Kinematic Model	19
2.4 Dynamic Model	24
3. SIMULATION	27
3.1 Introduction	27
3.2 Simulation Algorithm	30
3.3 Adjustment Blocks	33
3.4 Other Dynamic Characteristics	44
4. THE PROGRAM MODULES	47
4.1 Introduction	47
4.2 Program Modules	47
4.2.1 Block Matrix Operator	48
4.2.2 Simulator	55
4.2.3 Command Interpreter	60
4.2.4 Editor	66

4.3	Setting input data	67
4.4	Running the program	69
5.	RESULTS	70
5.1	Results	70
5.2	Conclusions	70
5.3	Suggestions	72
6.	REFERENCES	73
7.	APPENDIX	75
A 1	Transactions during the program run	75

## LIST OF SYMBOLS

N	Number of links of an Articulate Mechanical System (AMS)
$L_i$	$i$ th link of the AMS
$T_i$	$i$ th joint of the AMS
$G_i$	Centre of gravity of $L_i$
$C_i$	Body fixed coordinate frame of $L_i$
$O_i$	Origin of $C_i$
$a$	Link length
$\alpha$	Link twist
$\theta$	Joint angle
$u$	Joint distance
$p$	Position vector of the centre of gravity of the link
$m$	Mass of the link
$J$	Inertia tensor of the link
$A$	Transition block matrix
$B$	Jacobian block matrix
$q$	Internal coordinate vector
$X$	External coordinate vector
$P$	Drive vector
$F$	Force block vector
$M$	Moment block vector

## ABSTRACT

Dynamic performance is one of the most significant factors in the design of manipulators particularly for fast and accurate robots recently developed.

Until the end of the last decade, the choice of the manipulator , the actuator and the control system units was a subject of free speculation, frequently based on experience but lacking any systematic method. Hence the need for developing certain criteria and procedures for a systematic design of the manipulator. In the design phase, there is no efficient means except simulation to investigate and evaluate the highly non linear and coupled systems. The aim of this work is to create a software tool for the simulation of all the dynamical values and characteristics of manipulator operations in a particular task execution and thus permit a fast evaluation of a great number of different configurations.

## CHAPTER I

### INTRODUCTION

In 1920, the Czech writer Carel Kapek wrote a collective drama entitled R.U.R ( Rossum's Universal Robot ) and with its translation the word Robot entered the English vocabulary.

For half a century the word Robot appeared on the pages of science fiction stories and novels. Issac Assimov coined the name of the trade, ' Robotics ', and provided us Roboticians with an ethic. These terms were later adopted by the scientific and technical community.

The robot may be defined as a class of technical systems which imitate or substitute human locomotion or intellectual function. The Robot Institute of America gives a more precise and technical definition: ' A robot is a reprogrammable multifunctional manipulator designed to move materials, parts, tools, or specialised devices, through variable programmed motions for the performance of a variety of tasks'.

Although the size, structure, function and architecture of robots vary widely, all of them have certain essential features:

- a) Manipulator
- b) Controller
- c) Actuators
- d) Sensors

Robots possess several specific qualities in both a mechanical and a control sense. In the mechanical sense, a feature specific to manipulation robots is that they are open chain 'active mechanisms', in contrast to conventional mechanisms in which motion is produced primarily by the so called kinematic degrees of freedom. Also, they have a versatile structure, ranging from open to closed, from one to some other kind of boundary conditions. From a control viewpoint, robot systems represent, multi variable and essentially non-linear automatic control systems. A manipulation robot is also an example of dynamically coupled system, and the control task itself is a dynamic task. A further feature is the redundancy reflected in an excess of the degrees of freedom for producing certain functional movements of articulated mechanical system. While it permits a greater mechanical flexibility in task performance, it complicates the control system by introducing optimising procedures for solving the problem of system redundancy. The surplus dof are used to satisfy special additional requests.

A list of the uses of robots is impressive. Robots have been used for operation in dangerous environments, in oceanography, space exploration, agriculture, medicines, etc. In industry, their scope is virtually boundless. The development of automatically controlled industrial robots has witnessed three generations : Programmable, Supervisory Controlled, and Artificial Intelligence robots. The boom which we observe today in the sphere of industrial robots is not a mere tribute to vogue. These ' steel collar workers' are now handling, transforming, assembling and disassembling parts, tools and specialised systems. In contrast to hard automation, robots represent ' soft ' automation - flexible, neither product, nor operations, nor industry limited, and immune to obsolescence.

The distance separating Carel Kapek from Cybernetics was covered in a quarter of a century. The next twenty five year period built the foundations of what seemed pure fantasy. In this quarter we are witnessing a quantitative accumulation and a qualitative leap in the field of Robotry.

#### LITERATURE REVIEW

Although Robotics is a multidisciplinary field, incorporating the results of numerous scientific and engineering disciplines, two major areas can be identified: areas



involving mechanisms and areas involving control systems. In this work, we have confined ourselves only to the former: in particular to the dynamics of manipulators.

The first work in the field of dynamics of spatial mechanisms was published by N.G. Bruyevich [1] as far back as in 1937.

In 1963, H.J. Fletcher, L. Rongved and E.Y.Yu [2] studied the motion of a satellite composed of two rigid bodies, connected by a universal joint, under the load due to gravitation.

In 1965, W.W. Hooker and G. Margulien [2] inspired by the preceeding work, studied the general case, where  $N+1$  bodies are connected by means of joints with one or two rotational degrees of freedom. Although this method was a significant advance in that it used matrix formalism, it was not able to obtain the matrices as functions of system state.

R.E. Roberson and J. Wittenburg [3] in their approach have defined the system of bodies as a graph, whereby the elaborated and known graph properties are used.

The method of W.W. Hooker [4] provides a new possibility of eliminating constraints but presents a serious problem in deriving the motion equations.

The method of P.W. Likins [■] proposed in 1971, is a more direct application of the preceeding method. Moreover, without reducing the problem's generality, it simplifies the numerical designation of the links in the chain.

J Wittenburg [4] generalised the method to systems having the structure of a topological branch whose joints permit  $r$  revolute dof's and  $t$  linear dof's. It does not however, include fully closed chains, nor does it offer an explicit matrix procedure for obtaining the equations.

In 1968, J.J. Uicker [3] proposed a method based on Lagrange's equations for the study of dynamical behaviour of joint connected systems of arbitrary structure. The method is sufficiently general to enable motion equations to be simulated on a computer. To overcome certain deficiencies in this method, M. Renaud [■] proposed a method based on matrix calculus. In 1977, J. Zabala [5] proposed an algorithm based on M. Renaud's method for automatically formulating motion equations.

In 1974, E.P. Popov et al [■] proposed an algorithm for solving the inverse problem based on the Gauss principle of numerical minimisation. The direct problem is solved from the necessary conditions of minimum.

In 1979, M.Vukobratovic and V. Potkonjak [1] proposed a method based on Appel's equations, and is computer oriented to a great extent.

All the methods based on the general theorem of dynamics ( except the method of P.W. Likins ) are based on direct enumeration. However for an efficient computer solution, a recursive formulation is desirable. The methods using Newton's and Euler's methods are in principle complex due to the complexity of eliminating the constraints by forces and moments. Lagrangian equations provide the possibility of directly regarding the equations as functions of system control inputs. However, the inherent unsuitability of applying these equations lies in the need to calculate partial derivatives of Lagrangian functions.

#### OUTLINE OF THE PRESENT WORK

Modelling of the dynamics of Open Chain Active Mechanisms ( OCAM ) is a central point in the modern approach to manipulator design for two main reasons. One is connected with dynamic control, and the second is the development of procedures for optimal design of manipulators.

As a rule, the methods for the dynamic analysis of active mechanisms use generalised coordinates. However,

in practice such a solution is insufficient, because one needs to consider the so called functional robot motion. This is a motion satisfying certain practical demands. It is therefore necessary to obtain drives producing these functional motions. Considerations of functional dynamics are closely connected with control and conversly. Hence, the term Dynamic Control, meaning, control based on detailed knowledge of the system dynamic characteristics has been introduced.

OCAM have complicated behaviour including interaction among multiple joints, non linear effects, and varying inertia depending upon the arm configuration. In practice until the end of the last decade, the designer proceeded with the design of each joint mechanism without knowing the actual characteristics of the multi dof motion. Also, the choice of the kinematic scheme and its different parameters, the actuator units, and the control systems was a subject of free speculation, frequently based on experience but lacking any system of method. Hence many parameters, and often the motors were over powered. Hence the need for developing certain criteria and procedures for a systematic choice of manipulator configuration. In the design phase there is no efficient means except simulation to investigate and evalulate the highly non-linear and coupled systems.

The aim of this work is to develop an interactive program for the simulation of all the dynamical values and characteristics of the manipulator operations in a particular task execution, permitting the designer to view the characteristics in their entirety, and to evaluate a great number of different configurations as fast as possible.

The program consists of four main parts. The first part defines all the vector, matrix and block matrix operations necessary for the simulation algorithm.

The second part is the simulation algorithm. The algorithm automatically forms and solves the mathematical model of the robot dynamics using the block matrix method. The inputs for the algorithm are the manipulator configuration, the initial state and the manipulation task. The output of the algorithm are the drives in the joints, the P-N diagram, the energy consumed for the performance of the task, etc.

The third part is the Editor. It is used to perform interactive editing of the task, state or configuration of the manipulator.

The fourth part, the Command Interpreter, interprets the commands given by the user, demands the data to be provided, if necessary, then selects the desired primitives

and executes it. For the prevention of the user's careless errors, the program traps them at various places and issues the relevant messages.

The main program is written in Pascal. The subroutines for the drawing of graphs on the graphics terminal is written in Fortran. For Graphic action, PLOT-10 Interactive Graphics Library (IGL) is used.

## CHAPTER II

## MODELLING

## 2.1 INTRODUCTION:

The algorithms for modelling Open Chain Active Mechanisms (OCAM) can generally be classified into two groups : analytical and computer oriented. Analytical procedures have appeared before Computer Aided (CA) methods and originate within the multibody satellite dynamics. Unfortunately the application of these methods ' by hand ' is very tedious and are subject to mistakes. Also these algorithms include either unsuitable numerical operations or impose some calculations which are not formalised. For these reasons it becomes necessary to develop CA methods for mathematical modelling. It will enable the designer to analyse a number of different configurations and choose the most appropriate one to the future of the device. The development of CA methods which perform real time calculations of robot dynamics is a direct contribution to the synthesis of control algorithm for practical purposes.

Any CA method must satisfy the following requirements:

- a) The input data for the algorithm are : robot configuration, robot state, workspace state, and robot task. Using such input data the computer itself

forms and solves the mathematical model, i.e., robot dynamics. In principle three problems of dynamics may be solved : a direct and an inverse one, or a combination of these two.

- b) The algorithm includes no numerical differentiation.

The present CA methods may be divided into three groups:

- a) Methods based on the general theorem of dynamics and the Newton-Euler equations,
- b) Methods based on the second order Lagrange equations.
- c) Methods based on Gibbs-Appel equations and the Gauss principle.

This program uses the general theorem of dynamics and incorporates the Block Matrix method. The Block Matrix method represents the analytically derived mathematical model but by using suitable block formalism the model reduces to the compact matrix form suitable for solving on a computer.

The aim of each CA method is to derive functions  $f$  and  $g$  such that

$$\ddot{u} = f(u, \dot{u}, P, \text{mechanism configuration})$$

$$P = g(u, \dot{u}, \ddot{u}, \text{mechanism configuration})$$



where  $u$  is the generalised coordinate vector and  $P$  is the drive vector.

The functions  $f$  and  $g$  are not some explicitly prescribed or derived function. They represent large computation algorithms. The realisation of the algorithms  $f$  and  $g$  are specific to and characteristic of each method and it depends on the mechanical approach. The algorithms  $f$  and  $g$ , although mutually inverse, ( $f$  represents the direct problem and  $g$  the inverse one) are sometimes realised in rather different ways. Most of the CA methods consider the fifth class kinematic pairs only, i.e., joints permitting only one degree of freedom between the segments. If a compound joint is in question, then it is dissembled into a sequence of fifth class joints with small parameter segments between them.

The following assumptions have been made while forming the mathematical model:

- a) All the links are rigid bodies.
- b) All the articulations are fifth class pairs.
- c) All the articulations are perfect.
- d) There are not kinematic singularities.

The overall model is developed in three phases: Geometric, Kinematic, and Dynamic.

## 2.2 THE GEOMETRIC MODEL:

The kinematic structure of the arm has been modelled based on the Denavit and Hartenberg convention [1].

Referring to the Figure 2.1, the links of the Articulated Mechanical System (AMS) are numbered from zero at the base to  $N$  at the End Effector (EE). Links  $L_{i-1}$  and  $L_i$  are connected at joint  $T_i$ . The axis of a joint is defined as the axis of rotation for a revolute joint and as a line parallel to the generatrix for a prismatic joint. A convenient fixed coordinate system is chosen for the AMS. A Body Fixed (BF) coordinate system  $C_i$  is attached to each link  $L_i$ . The origin of frame  $C_i$  is denoted by  $O_i$ . The  $z$  axis of the frame  $C_i$  is oriented along the axis of the joint,  $T_i$  and the  $x$  axis is directed along the common normal. The  $y$  axis is so chosen as to make a right handed coordinate system. The relative position of successive pair axes is described by the use of the unique common perpendicular between by axes of successive coordinate frames. With each link  $L_i$  is associated a vector  $\rho_i$  defining the centre of mass of  $L_i$  with respect to the frame  $C_i$ , a mass  $M_i$  and an inertia tensor  $J_i$ .

Four parameters are used to describe each successive joint and link pair:

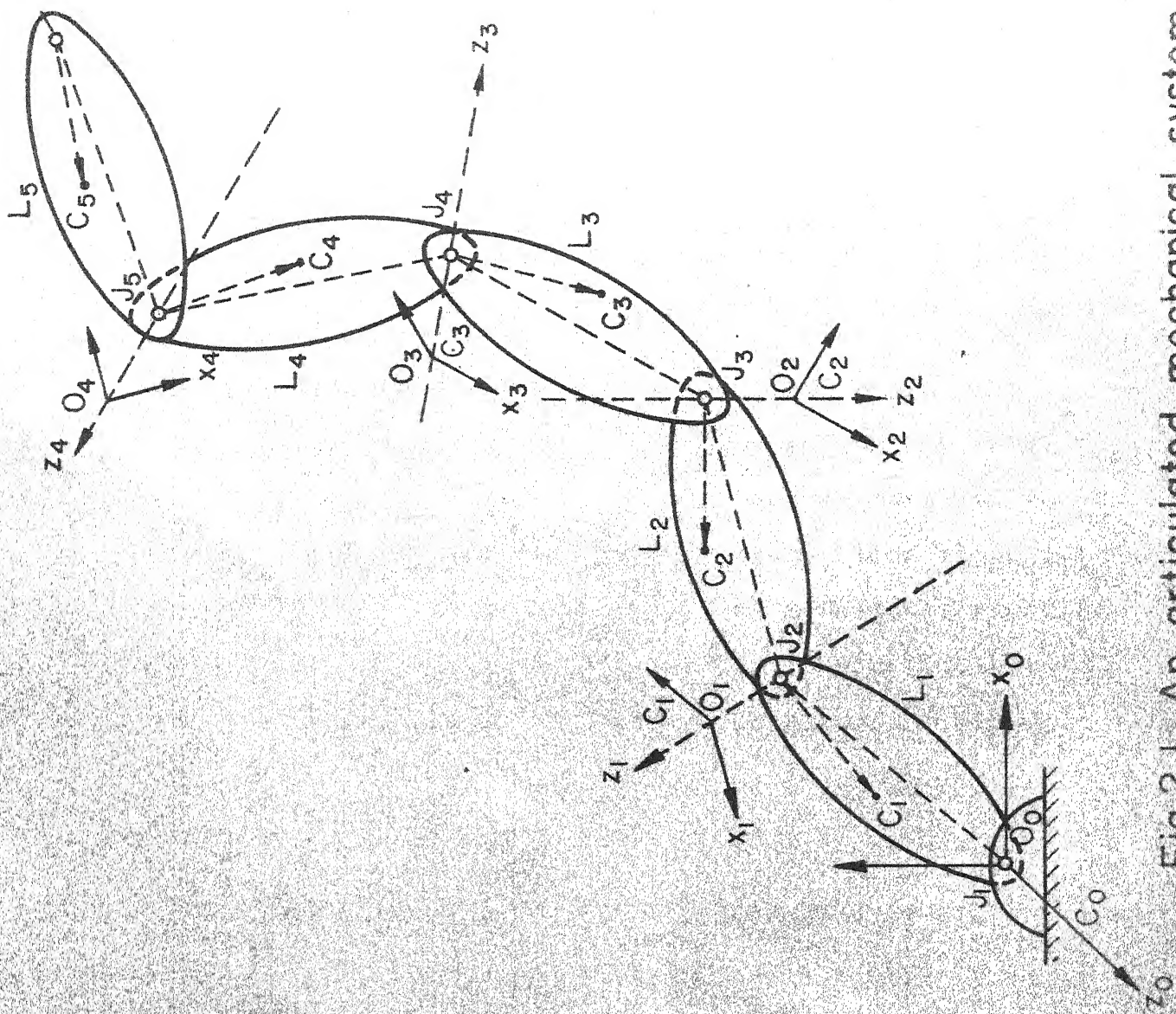


Fig.2.1 An articulated mechanical system

Link length,  $a$  - distance between successive  
z axes

Link twist,  $\alpha$  - angle between successive  
z axes

Joint angle,  $\theta$  - angle between successive  
x axes

Joint distance,  $u$  - distance between successive  
x axes.

Referring to Fig. 2.2, the transformation of the frame  $C_{i-1}$  into the frame  $C_i$  may be represented by the concatenation of the following elementary transformations:

- a)  $R(z_{i-1}, \theta_i)$  - rotation around  $z_{i-1}$   
by an angle  $\theta$  until  $x_{i-1}$   
becomes parallel to  $x_i$
- b)  $T(z_{i-1}, u_i)$  - translation along  $z_{i-1}$   
by a distance  $s$  until  
 $x_{i-1}$  coincides with  $x_i$
- c)  $T(x_i, a_i)$  - translation along  $x_i$   
by a distance  $a$  until  
 $O_{i-1}$  coincides with  $O_i$
- d)  $R(x_i, \alpha_i)$  - Rotation around  $x_i$  by an  
angle  $\alpha$  until  $C_{i-1}$  and  $C_i$   
coincide.

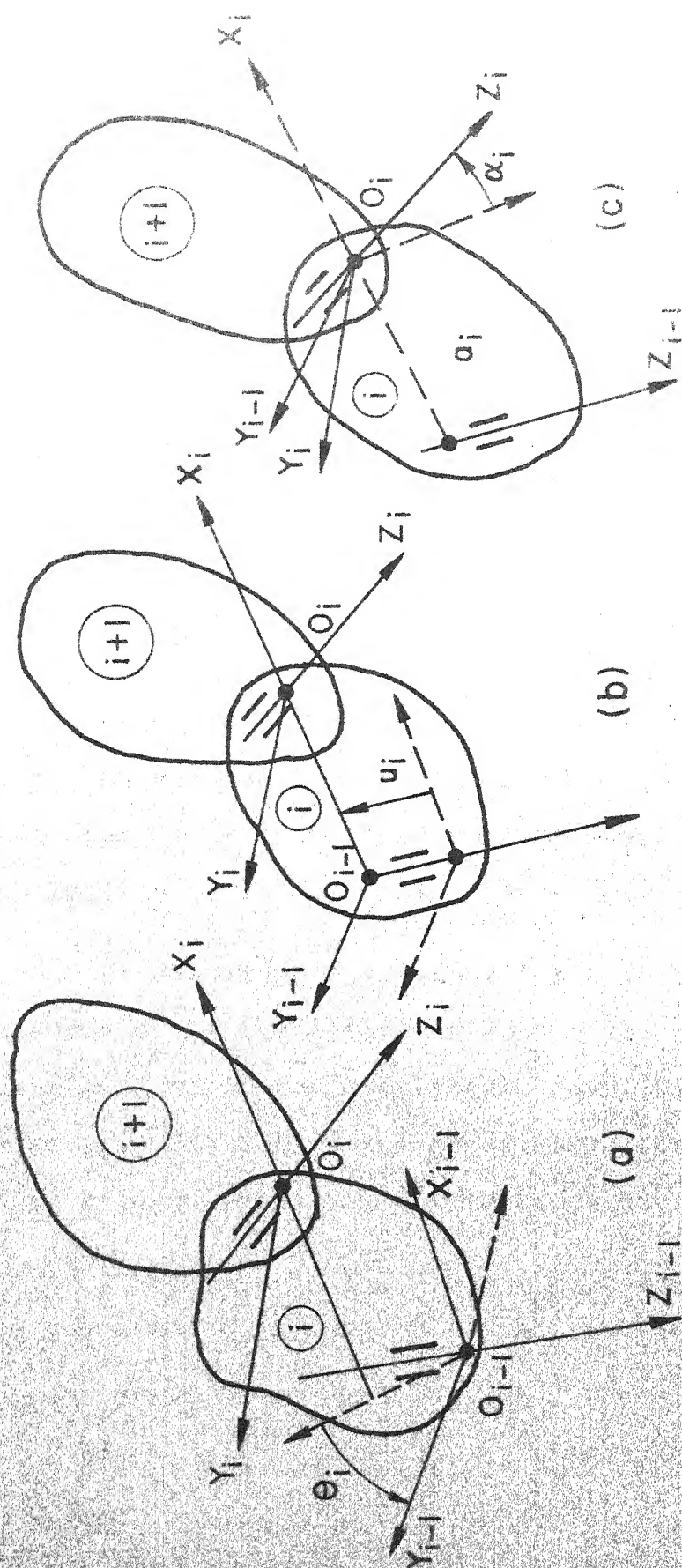


Fig.2.2 Illustrating transition from  $(i-1)$ st to  $i^{\text{th}}$  frame of reference

From these four elementary transformations, a 4 X 4 homogenous transformation matrix  $T_{i-1,i}$ , which relates the position and orientation of frame  $C_i$  to that of frame  $C_{i-1}$  may easily be formed.

$$T_{i-1,i} = \begin{bmatrix} \cos \theta_i & \sin \theta_i & 0 & a_i \cos \theta_i \\ -\sin \theta_i \cos \alpha_i & \cos \theta_i \cos \alpha_i & \sin \alpha_i & a_i \sin \theta_i \\ \sin \theta_i \sin \alpha_i & -\cos \theta_i \sin \alpha_i & \cos \alpha_i & u_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \dots (2.1)$$

Here,  $T_{i-1,i}$  is an orthogonal matrix as it transforms one orthogonal system into another. The transformation between other link coordinate systems may be obtained by the multiplication of intermediate transformation matrices.

It should be noted that for a revolute joint, the joint angle  $\theta_i$  is the joint variable ( the generalised coordinate ), while for a prismatic joint, it is the joint distance  $u_i$ . Similarly, the generalised force for a revolute joint is a torque, and for prismatic joint, it is a force.

A point  $r^p$  in the frame  $C_p$  transforms to the point  $r^q$  in the frame  $C_q$  according to the equation

$$r^p = T_{p,q} \cdot r^q \quad \dots (2.2)$$

Let  $A_{i-1,i}$  be a 3 X 3 sub matrix of  $T_{i-1,i}$  giving the orientation of the axes of  $C_{i-1}$  with reference to  $C_i$

$$A_{i-1,i} = \begin{bmatrix} \cos \theta_i & \sin \theta_i & 0 \\ -\sin \theta_i \cdot \cos \alpha_i & \cos \theta_i \cdot \cos \alpha_i & \sin \alpha_i \\ \sin \theta_i \cdot \sin \alpha_i & -\cos \theta_i \cdot \sin \alpha_i & \cos \alpha_i \end{bmatrix} \quad \dots (2.3)$$

And  $l_{i-1,i}$  be 3 X 1 vector giving the position of the origin of the frame  $C_{i-1}$  with reference to  $C_i$

$$l_{i-1,i} = [a_i \cdot \cos \theta_i \quad a_i \cdot \sin \theta_i \quad u_i] \quad \dots (2.4)$$

The equation 2.2 may also be written as

$$r^{i-1} = A_{i-1,i} r^i + l_{i-1,i} \quad \dots (2.5)$$

Then for a vector  $V_i$ ,

$$V_i^p = A_{p,q} V_i^q \quad \dots (2.6)$$

where  $A_{p,q}$  is obtained by the multiplication of intermediate transformation matrices

$$A_{p,q} = A_{p,p+1} \cdot A_{p+1,p+2} \cdot \dots \cdot A_{q-1,q}$$

$A_{0,i}$  is written simply as  $A_i$  and  $l_{0,i}$  as  $l_i$

Applying equations 2.5 and 2.6 repeatedly, we get

$$r_i = \sum_{j=0}^i A_j l_{j,j} + A_i r_i^i \quad \dots(2.7)$$

$$V_i = A_i V_i^i \quad \dots(2.8)$$

The three columns of the matrix  $A_N$  represents the orientation of the axes of the frame  $C_N$  in terms of the generalised coordinates, and the vector  $l_N$  represents the position vector of the centre of gravity of the EE in the external coordinate system. Thus the geometric model enables us to convert internal coordinates to external coordinates.

### 2.3 KINEMATIC MODEL:

The derivation of the equations constituting the kinematic and dynamic model is very lengthy and complex, and is discussed in [6]. The analytical equations themselves are very long. However, by using block matrix formalism, the equations are reduced to a very compact form. The equations, are presented here only in their final form [7].

Let  $a_1, \dots, a_N$  be a set of vectors, and let us define the Block vectors  $a$  and  $a^0$  of dimension (  $N \times 1$  (  $3 \times 1$  ) )

$$a = [ a_1^1, \dots, a_N^N ]^T$$

$$a^0 = [ a_1 \dots a_N ]^T$$

Let  $E_3$  be a  $3 \times 3$  unit matrix, and  $E_N$ , a  $N \times N$  unit matrix let  $V$  be the (  $N \times N$  (  $3 \times 3$  ) ) block matrix.



$$V = \begin{bmatrix} E_3 & 0 & \dots\dots 0 \\ E_3 & E_3 & \dots\dots 0 \\ \vdots & & & \\ E_3 & E_3 & \dots\dots E_3 \end{bmatrix}$$

Further, let us define the link type matrix,  $S (N \times N)$ , and the block matrix  $\mathcal{V} (N \times N (3 \times 1))$  as follows

$$\begin{aligned} S &= \text{diag} [s_1 \ s_2 \ \dots\dots s_N] \\ \mathcal{V} &= \text{diag} [v_1 \ v_2 \ \dots\dots v_N] \end{aligned} \quad \dots(2.9)$$

where  $v_i = [0 \ \sin \alpha_i \ \cos \alpha_i]^T$

and  $s_i$  is an indicator, whose value is 0 if the  $i$ th joint is revolute and 1 if it is prismatic.

Let  $A$  be the orientation block matrix  $(N \times N (3 \times 3))$

$$A = \begin{bmatrix} E_3 & 0 & 0 & \dots\dots 0 \\ A_{21} & E_3 & 0 & \dots\dots 0 \\ A_{31} & A_{32} & E_3 & \dots\dots 0 \\ \vdots & & & \\ A_{N1} & A_{N2} & A_{N3} & \dots\dots E_3 \end{bmatrix}$$

Now, let us define two functions  $\lambda$  and  $\Lambda$  as follows:

$$\lambda(V_i) = \begin{bmatrix} 0 & -V_{iz} & V_{iy} \\ V_{iz} & 0 & -V_{ix} \\ -V_{iy} & V_{ix} & 0 \end{bmatrix} \quad \dots(2.10)$$

where  $V_i = [V_{ix} \ V_{iy} \ V_{iz}]^T$

$$A(V) = \text{diag} [\lambda(V_1) \quad \lambda(V_2) \quad \dots \quad \lambda(V_N)]$$

where  $V_i$  is a vector (3X1)

Let  $q = [q_1 \ q_2 \ \dots \ q_N]^T$  be the vector of internal coordinates (N X 1) and  $X = [X_1 \ X_2 \ \dots \ X_N]^T$ , the vector of external coordinates (N X 1). If  $v$  is the linear velocity vector, and  $\omega$ , the angular velocity vector, then  $\dot{X} = [v \ \omega]^T$  is a (2 X 1, (3 X 1)) velocity block matrix.

Now, we have the following relations for velocity block matrix

$$\dot{X} = B\dot{q} \quad \dots (2.12)$$

where B is a (2 X 1 (N X N (3 X 1))) block matrix

$$B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

$$B_1 = -(\Lambda\Lambda(1) + \Lambda(\rho) \quad A v(E-S) + A v s \quad \dots (2.13)$$

$$B_2 = A v(E-S)$$

For velocities expressed in the external coordinate system, we have

$$\dot{X}^0 = B^0 \dot{q} \quad \dots (2.14)$$

where

$$B^0 = \begin{bmatrix} B_1^0 \\ B_2^0 \end{bmatrix} \quad \dots (2.15)$$

$$B_1^0 = -\Lambda^*(\rho) e^0 (E-S) \dot{q} + V S e^0 \dot{q}$$

$$B_2^0 = V e^0 (E-S) \dot{q}$$

$$e^0 = \text{diag} [e_1 \ \dots \ e_N]$$

$$\Lambda^* (\rho) = \begin{bmatrix} \lambda(\rho_{01}) & 0 & \dots & 0 \\ \lambda(\rho_{02}) & \lambda(\rho_{12}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda(\rho_{0N}) & \lambda(\rho_{1N}) & \dots & \lambda(\rho_{N-1,N}) \end{bmatrix}^{22}$$

Now, let  $w$  and  $\epsilon$  be linear and angular acceleration vectors (  $3 \times 1$  ) respectively. Then  $\ddot{X} = [w \ \epsilon]^T$

We have the following relation for acceleration block vector

$$\ddot{X} = B\ddot{q} + D\dot{q} \quad \dots(2.16)$$

where

$$D = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}$$

$$D_1 = (A\Lambda(1) + \Lambda(\rho)) A\Lambda(\omega) \nu(E-S - (A\Lambda(\omega)\Lambda(1) + \Lambda(\omega)\Lambda(\rho)) A\nu(E-S) + 2A\Lambda(\omega)\nu S) \quad \dots$$

$$D_2 = A(\Lambda(\omega) \nu(E-S)) \quad \dots(2.17)$$

In the external coordinate system,

$$\ddot{X}^0 = B^0\ddot{q} + D^0\dot{q} \quad \dots(2.18)$$

where

$$D^0 = \begin{bmatrix} D_i^0 \\ D_2^0 \end{bmatrix}$$

$$D_1^0 = [D_1^1 \dots D_N^1]^T \quad \text{dimension } (N \times N (3 \times 1))$$

$$D_i^1 = Q(D_{i1}^1 + D_{i2}^1) \quad \text{dimension } (3 \times N)$$

$$D_{il}^1 = \begin{bmatrix} 1 & \dots & d_{li}^1 & 0 & \dots & 0 \\ d_{11}^1 & & & & & \\ \vdots & & & & & \\ d_{i-1,i}^1 & & d_{i,1,i}^1 & 0 & \dots & 0 \\ 0 & \dots & & 0 & \dots & 0 \\ \vdots & & & & & \\ 0 & \dots & & 0 & \dots & 0 \end{bmatrix} \quad \dots (2.19)$$

$$d_{kj}^1 = \lambda(e_{j-1}) \lambda(e_k) \rho_{j-1,i} (1-s_k) (1-s_j)$$

$$\begin{bmatrix} 0 & \dots & d_{12}^2 & \dots & d_{li}^2 & \dots & 0 & \dots & 0 \\ d_{21}^2 & & 0 & \dots & d_{2i}^2 & & 0 & \dots & 0 \\ \vdots & & & & & & & & \\ d_{i1}^2 & & d_{i2}^2 & \dots & 0 & & 0 & \dots & 0 \\ \vdots & & & & & & & & \\ 0 & & 0 & & 0 & & 0 & \dots & 0 \end{bmatrix}$$

$$d_{kj}^2 = \lambda(e_{k-1}) e_{j-1} (1-s_k) s_j$$

$$d_{jk}^2 = d_{kj}^2$$

$$D_2^0 = [D_1^2 \quad \dots \quad D_N^2]^T \quad \text{dimension } (N \times N (3 \times 1))$$

$$D_i^2 = \dot{Q} H_i^2$$

$$\dot{Q} = [\dot{q}_1 E_3 \quad \dot{q}_2 E_3 \quad \dots \quad \dot{q}_N E_3]$$

$$H_i^2 = \begin{bmatrix} 0 & 0 & \dots & 0 & \dots & 0 \\ \sigma_1 \sigma_2 \lambda(e_0) e_1 & 0 & \dots & 0 & \dots & 0 \\ \sigma_1 \sigma_3 \lambda(e_0) e_2 & \sigma_2 \sigma_3 \lambda(e_1) e_2 & & 0 & & 0 \\ \sigma_1 \sigma_i \lambda(e_0) e_{i-1} & \sigma_2 \sigma_i \lambda(e_1) e_{i-1} \dots \sigma_{i-1} \sigma_i \lambda(e_{i-2}) e_{i-1} \dots & & 0 & & 0 \\ 0 & & & & & \\ \vdots & & & & & \\ \vdots & & & & & \\ \vdots & & & & & \\ \vdots & & & & & \\ \vdots & & & & & \\ 0 & 0 & \dots & 0 & \dots & 0 \end{bmatrix}$$

$$\sigma_i = 1 - s_i$$

We now have the expressions for the velocity and acceleration of the links in the external coordinate system as well as the BF system. These expressions constitute the kinematic model.

#### 2.4 DYNAMIC MODEL:

The dynamic equations are derived on the basis of D'Alambert's principle by interrupting the chain fictitiously at a joint and balancing the forces and moments. In this way, N scalar equations are obtained, and transformed into matrix form using block formalism.

Let us introduce the following block matrices of dimension (NXN (3X3)).

$$\begin{aligned}
 m &= \text{diag} [ m_1 E_3 \quad \dots \quad m_N E_3 ] \\
 J^0 &= \text{diag} [ J_1^0 \quad \dots \quad J_N^0 ] \\
 A_0 &= \text{diag} [ A_{10} \quad \dots \quad A_{N0} ]
 \end{aligned}$$

The expressions for the block vectors of the resultant inertial forces and moments can be written together as

$$\begin{bmatrix} F_I^0 \\ M_I^0 \end{bmatrix} = - \mathcal{J} B^0 q - C^0 q \quad \dots (2.20)$$

where

$$\begin{aligned}
 \mathcal{J} &= \begin{bmatrix} m & 0 \\ 0 & J^0 \end{bmatrix} \\
 C^0 &= \begin{bmatrix} m D_1^0 \\ \Omega^0 J^0 B_2^0 + J^0 D_2^0 \end{bmatrix} \\
 \Omega &= \Lambda (\omega^0) \quad \dots (2.21)
 \end{aligned}$$

Let us further introduce the vector of the drives  $P$  of dimension  $(N \times 1)$

$$B = [ P_1 \quad \dots \quad P_N ]^T$$

Then the dynamic equations can be written as

$$W \ddot{q} = B' \dot{q} + C' M_E^0 + D' (G^0 + F_E^0) + P \quad \dots (2.22)$$

where

$$\begin{aligned} W &= B^0 T B^0 \\ B' &= B^0 T C^0 \\ C' &= B_2^0 T \\ D' &= B_1^0 T \end{aligned} \quad \dots (2.23)$$

by introducing

$$U(q, \dot{q}) = B' \dot{q} + C' M_E^0 + D' (G^0 + F_E^0) \dots (2.24)$$

the equation acquires the form

$$W(q) \ddot{q} = U(q, \dot{q}) + P \quad \dots (2.25)$$

Thus we have described the dynamics of the AMS by means of N differential equations in matrix form.

## CHAPTER III

## SIMULATION

## 3.1 INTRODUCTION:

The notion of the simulation of dynamics usually involves the solution of the inverse problem of dynamics, i.e. the determination of the motion for prescribed generalised forces. In this case, the simulation is considered somewhat more liberally so that it also includes the notion of the simulation of the direct problem.

The Fig. 3.1 shows the different models and the different levels of control of a typical manipulator organised in a hierarchial fashion. The highest level defines the task to be carried out, the strategic level divides the imposed task into elementary functions, the tactical level performs the distribution of an elementary movement to the motion of each degree of freedom of the robot, and the executive level executes the imposed motion of each dof. The model developed in the previous chapter is used as the basis for the development of the algorithm for the simulation of the manipulator dynamics, and thus the synthesis of the tactical level.

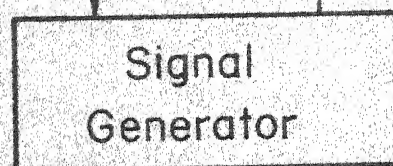
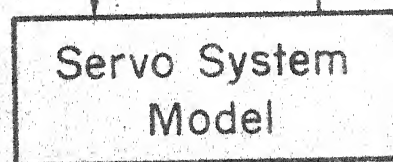
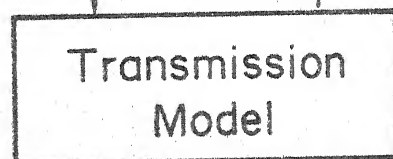
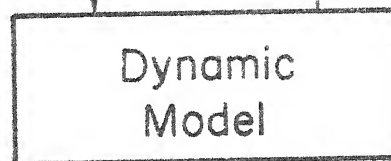
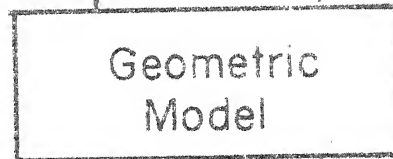
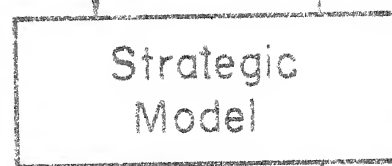
Before proceeding further, it will be useful to note an important point. The task of a manipulator may



Strategic Level

Tactical Level

Executive Level



Command

 $X(t)$  $q(t)$  $D(t)$  $T(t)$  $V(t)$ 

Fig. 3.1 The Different Control Levels And Model of a Typical Manipulator.

be defined as the process of transferring the system state from one bounded region of initial states into another bounded region in the state space within a finite settling time, the system state being in the bounded region of the state space during the transfer. The state of the Articulated Mechanical System (AMS), at any instant may be described by means of a vector of coordinates. The coordinates are called internal coordinates if the elements of the vector represent the position of the articulation at that time instant. If they represent the coordinates of a predefined point on the End Effector (EE) with reference to the external coordinate system, then they are called the external coordinates. However, in practice, the task is prescribed neither by means of internal coordinates, nor the external coordinates, but by the so called functional coordinates, as it is necessary to consider the functional robot motion. This is a motion satisfying certain practical demands. For instance, for a particular operation, the task of positioning the minimal configuration may be given in terms of spherical coordinates, and the task of orientation of the the EE may be given in terms of three Euler angles. For another operation it may be more convenient to give the orientation of the EE in terms of a direction and an angle of rotation round it.

Transformation of the functional coordinates into external coordinates is fairly simple, but the transformation of the external coordinates to internal coordinates is extremely complicated not only because the transformation is not explicit, but because it cannot even be numerically approximated due to the complexity of the system. Also, a set of external coordinates may not have a unique set of corresponding internal coordinates. The external coordinates define the position and orientation of the EE, which we may call 'pose' of the manipulator. On the other hand the internal coordinates determine the 'posture' of the manipulator, i.e., the geometrical configuration adopted by the manipulator while holding an object in a particular pose. The posture of a manipulator completely defines its pose, but for a particular pose a manipulator may be in different postures. A 3 dof manipulator has only one posture in general, and a 6 dof one is believed to have 32 postures in general. A human arm can adopt an infinite number of postures.

### 3.2 THE SIMULATION ALGORITHM:

Let us designate by  $\eta$  a function which transforms the internal coordinates into external coordinates.

$$X = \eta(q) \quad \dots\dots\dots (3.1)$$

Now, for the operation of the Computer Aided (CA) method of forming the mathematical model, it is necessary to know  $q$ ,  $\dot{q}$  and  $\ddot{q}$  at each time instant. However, only  $q$  appears as input since  $\dot{q}$  and  $\ddot{q}$  are calculated by integration starting with known initial state  $q^0$  and  $\dot{q}^0$ . So in order to realise the simulation, it is necessary to develop a procedure for calculating  $\ddot{q}$  from the known state  $q$ ,  $\dot{q}$ , and known external coordinate vector,  $X$ .

By differentiating equation 3.1 twice,

$$\dot{X} = \frac{dn}{dq} \cdot \dot{q} \quad \dots (3.2)$$

$$X = \frac{dn}{dq} \cdot \dot{q} + \frac{d^2n}{dq^2} \cdot \dot{q}^2 \quad \dots (3.3)$$

Let  $B = \frac{dn}{dq}$ ,  $A = \frac{d^2n}{dq^2} \cdot \dot{q}^2$  Then

$$\ddot{X} = B \ddot{q} + A \quad \dots (3.4)$$

$B$  is called the Jacobian matrix (order  $N \times N$ ).

$B$  and  $A$  are functions of the state  $q$ ,  $\dot{q}$ , and are calculated numerically by means of the kinematic model developed in the previous chapter. Then  $\ddot{q}$  is calculated by the relation

$$\ddot{q} = B^{-1} (\ddot{X} - A) \quad \dots (3.5)$$

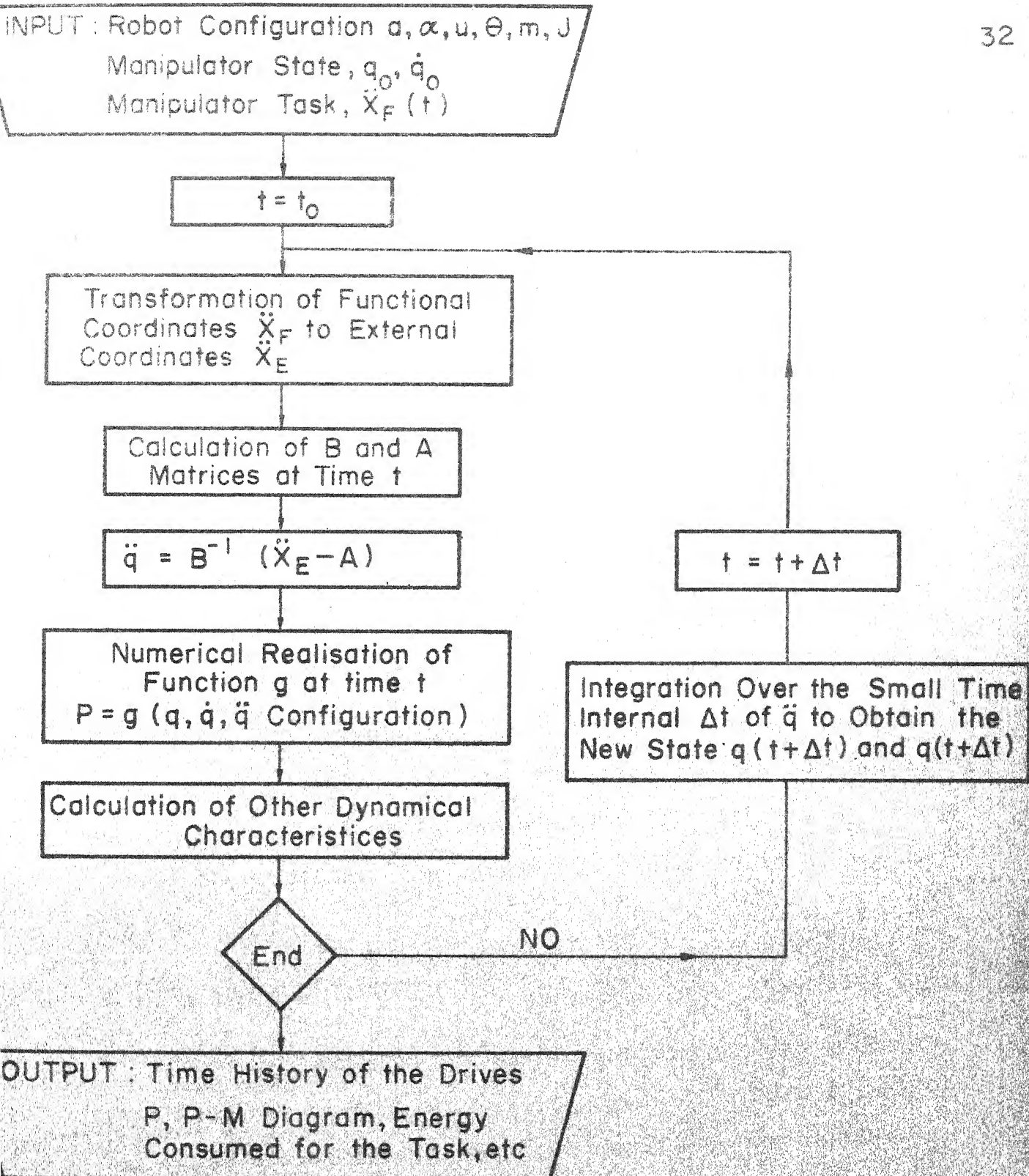


FIG. 3.2 FLOW CHART OF THE SIMULATION ALGORITHM .

$\ddot{X}$  can be represented by

$$\ddot{X} = \begin{bmatrix} w \\ \epsilon \end{bmatrix}$$

where  $w$  is the linear acceleration, and  $\epsilon$  is the angular acceleration of the centre of gravity of the EE.

From Eqn. 2.16, we can obtain

$$\begin{aligned} \dot{X} &= B\dot{q} + D\dot{q} \\ &= B\ddot{q} + A \end{aligned} \quad \dots(3.6)$$

Now from Eqn. 2.25 we can calculate the drives

$$\begin{aligned} P &= g(q, \dot{q}, \ddot{q}, \text{configuration}) \\ &= W\ddot{q} - U \end{aligned} \quad \dots (3.7)$$

By integrating  $\ddot{q}$ , we get the state  $q, \dot{q}$  at the next time instant. With the new state and the corresponding value of the functional coordinates acceleration,  $X_F$  we can repeat the process. The simulation algorithm is presented in Fig. 3.2.

### 3.3 ADJUSTMENT BLOCKS :

As pointed out earlier, although the manipulation task can be defined fully by means of external coordinates, they are unsuitable for setting. For the various individual classes of tasks, the most suitable variables are given as inputs, which naturally reflect the type of the

task. These are called the, functional coordinates. The transformation of the functional coordinates to the external coordinates is done in a block, known as the adjustment block, within the simulation algorithm.

Before proceeding with the various classes of tasks, let us define some notions precisely. By positioning we mean moving the centre of gravity of the last segment ( or some predefined point on it ) to some desired point in the work space according to a prescribed motion law.

Full orientation of the body in space means an exactly determined angular position of the body with respect to the external space.

Partial orientation of the body means that the given body axis coincides with a prescribed direction in space.

To solve the positioning task, which is part of every manipulation task, three dof are necessary.

To solve the positioning task along with the task of partial orientation, five dof are necessary.

To solve the positioning task along with the task of full orientation, six dof are needed.

Now some typical classes of tasks and ways of using the dof will be analysed.

A) A manipulator with four dof solves the positioning task by using three dof, and with the one remaining performs operations frequently sufficient for many practical tasks.

1. The task is that of positioning in the cartesian coordinate system, i.e.  $x(t)$ ,  $y(t)$ ,  $z(t)$  for the centre of gravity of the EE, and the fourth dof is prescribed directly ( $q_4(t)$ ).
2. Same as 1., but positioning is given in cylindrical coordinates  $(t)$ ,  $\theta(t)$ ,  $z(t)$ .
3. Same as 1., but positioning is given in spherical coordinates  $r'(t)$ ,  $\theta(t)$ ,  $\beta(t)$ .

B) A manipulator with five dof solves the positioning and partial orientation task.

1. The task is given in the form of positioning ( $A_1$ ,  $A_2$  or  $A_3$ ) and partial orientation.

C) A manipulator with six dof solves the positioning and full orientation task, as well as all the problems in which fewer dof are needed (when compensation of the dof surplus is done).



1. The task is given in the form of positioning (A1, A2, or A3), and full orientation of the EE in terms of the three Euler angles,  $\Theta(t)$ ,  $\psi(t)$ ,  $\beta(t)$ .
2. The task is given in the form of positioning (A1, A2, or A3) and full orientation is given in terms of one direction  $\Theta(t)$  and  $\beta(t)$ , and an angle of rotation around it,  $\psi(t)$ .
3. The task is given in terms of positioning as in 2 plus partial orientation. Motion along one dof is prescribed directly ( $q_k(t)$ ).

The derivation of the adjustment blocks for these classes of task will now be given briefly. We will use the same of matrix notation for  $X_E$  (external coordinates) i.e.

$$\ddot{X}_E = \begin{bmatrix} w \\ \epsilon \end{bmatrix} = B\ddot{q} + A$$

Let  $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$  and  $A = \begin{bmatrix} 0 \\ \phi \end{bmatrix}$

A1 The input to the block is  $X_F$  (functional coordinates)

$$\ddot{X}_F = [\ddot{x} \ \ddot{y} \ \ddot{z} \ \ddot{q}_4]^T$$

Hence,  $w = [\ddot{x} \ \ddot{y} \ \ddot{z}]^T \dots (3.7)$

Using the equation 3.5  $\ddot{q}_1, \ddot{q}_2$  and  $\ddot{q}_3$  are calculated  $\ddot{q}_4$  is prescribed directly.

$$\underline{A2} \quad \ddot{X}_F = [\rho \ddot{\theta} \dot{z} \ddot{q}_4]^T$$

The following equations can easily be derived using elementary transformation.

$$\begin{aligned} w_x &= w_\rho \cos \theta - w_\theta \sin \theta \\ w_y &= w_\rho \sin \theta + w_\theta \cos \theta \\ w_z &= \dot{z} \end{aligned} \quad \dots (3.8)$$

$$\text{where } w_\rho = \ddot{\rho} - \rho \dot{\theta}^2, \quad w_\theta = \rho \ddot{\theta} + 2\dot{\rho} \dot{\theta}, \quad w_z = \ddot{z}$$

$\rho, \dot{\rho}, \theta, \dot{\theta}$  etc are obtained by integration of  $\rho, \ddot{\theta}$  etc. the rest is as in A1.

$$\underline{A3} \quad \ddot{X}_F = [\ddot{r}, \ddot{\theta}, \ddot{\beta}, \ddot{q}_4]^T.$$

Again, the following equations can easily be derived.

Please refer to Fig. 3.3. a,b,c.

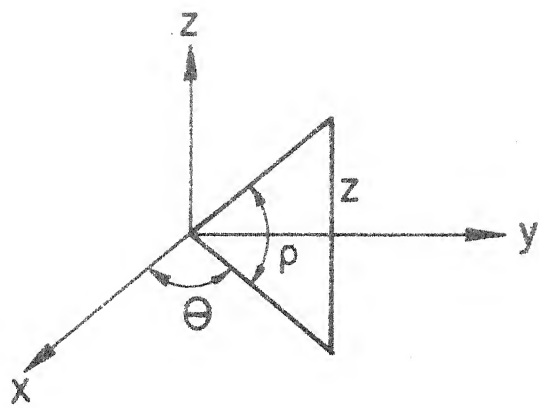
$$\begin{aligned} w_x &= w_r \cos \theta \cos \beta - w_\theta \sin \theta - w_\beta \sin \theta \cos \beta \\ w_y &= w_r \cos \theta \sin \beta + w_\theta \cos \theta - w_\beta \sin \theta \sin \beta \\ w_z &= w_r \sin \theta + w_\beta \cos \theta \end{aligned} \quad \dots (3.9)$$

where

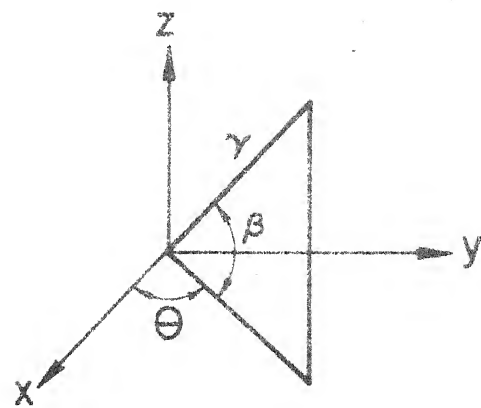
$$\begin{aligned} w_r &= \ddot{r} - r \dot{\beta}^2 - r \cos^2 \beta \dot{\theta}^2 \\ w_\theta &= 2\dot{r}\dot{\theta} \cos \beta + r\ddot{\theta} \cos \beta - 2r\dot{\beta}\dot{\theta} \sin \beta \\ w_\beta &= 2\dot{r}\dot{\beta} + r\ddot{\beta} + r \sin \beta \cos \beta \dot{\theta}^2 \end{aligned}$$

C1 If positioning is prescribed in cartesian coordinates,

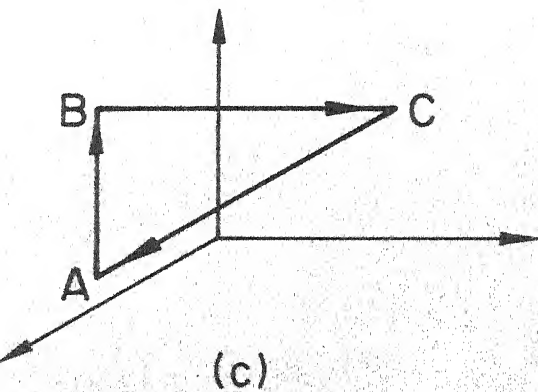
$$\ddot{X}_F = [\ddot{x} \quad \ddot{y} \quad \ddot{z} \quad \ddot{\theta} \quad \ddot{\psi} \quad \ddot{\beta}]^T.$$



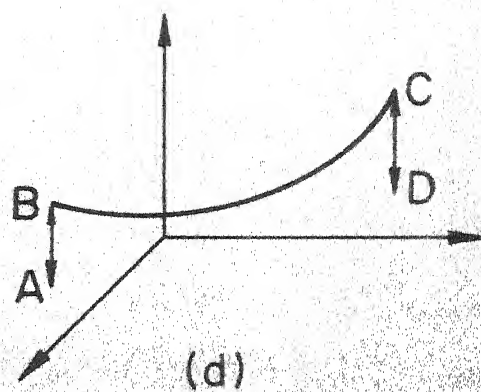
(a) Cylindrical co-ordinates



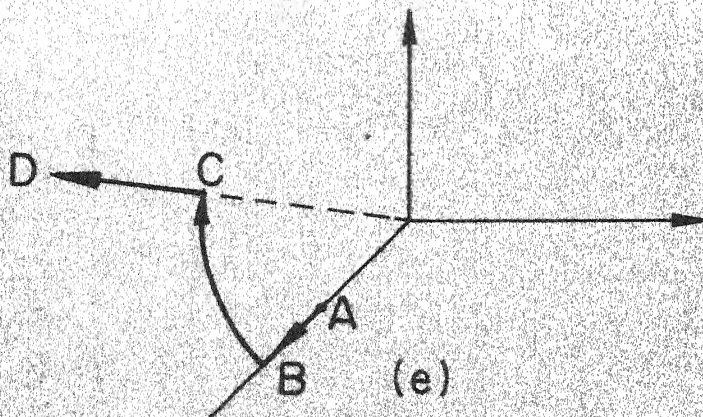
(b) Spherical co-ordinates



(c)



(d)



(e)

Fig.3.3 Various trajectories of the manipulator tip

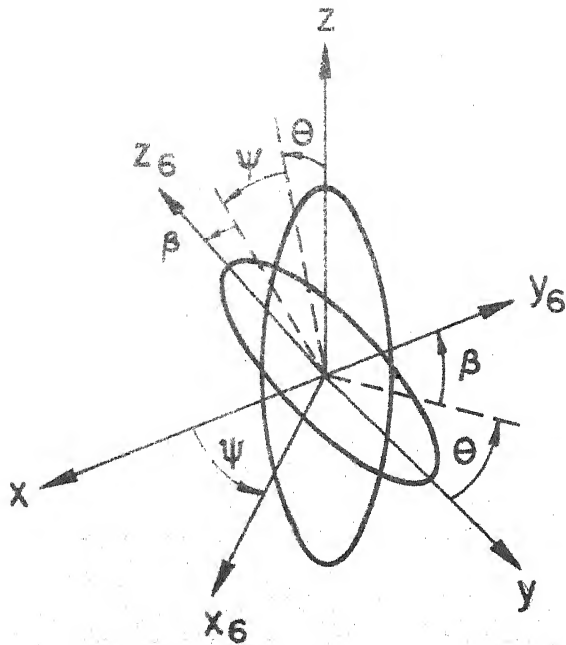


Fig.3.4 Euler angles of the BF system ( $x_6, y_6, z_6$ ) relative to the external system ( $x, y, z$ )

The positioning may also be prescribed in cylindrical or spherical coordinates. In any case,  $w$  is calculated as in A1, A2, or A3, as required. Please refer to Fig. 3.4.

The angular acceleration is given by

$$\epsilon = A_6 \left( \pi \begin{bmatrix} \ddot{\theta} \\ \dot{\psi} \\ \dot{\beta} \end{bmatrix} + {}^1\pi \begin{bmatrix} \dot{\psi} \\ \dot{\beta} \\ \dot{\theta} \end{bmatrix} \right)$$

where

$$\pi = \begin{bmatrix} \cos \psi & 0 & 1 \\ \sin \psi \cdot \sin \beta & \cos \beta & 0 \\ \sin \psi \cdot \cos \beta & -\sin \beta & 0 \end{bmatrix}$$

$${}^1\pi = \begin{bmatrix} 0 & 0 & -\sin \psi \\ -\sin \beta & \sin \psi \cdot \cos \beta & \cos \psi \cdot \sin \psi \\ -\cos \beta & -\sin \psi \cdot \sin \beta & \cos \psi \cdot \cos \beta \end{bmatrix} \quad \dots (3.10)$$

$A_6$  is the transition matrix of the last segment

$$A_6 = A_1 A_2 \dots A_6.$$

C2 This method is suitable in many practical cases, for eg. spraying powder along a prescribed path, or screwing in a bolt.

$$\ddot{X}_F = [\ddot{x} \quad \ddot{y} \quad \ddot{z} \quad \ddot{\theta} \quad \ddot{\beta} \quad \ddot{\psi}]^T$$

$w$  is determined as in C1.

Referring to Fig. 3.5, the direction of any line may be prescribed by means of two angles  $\theta$  and  $\beta$ . Rotation around this line is given by  $\psi$ .

$\epsilon$  is calculated from the relation

$$\lambda(\epsilon) = \ddot{A} A^{-1} - [\lambda(\omega)]^2 \quad \dots (3.11)$$

where

$$A = A_1 A_2 A_3$$

$$A_1 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi \\ 0 & \sin \psi & \cos \psi \end{bmatrix}$$

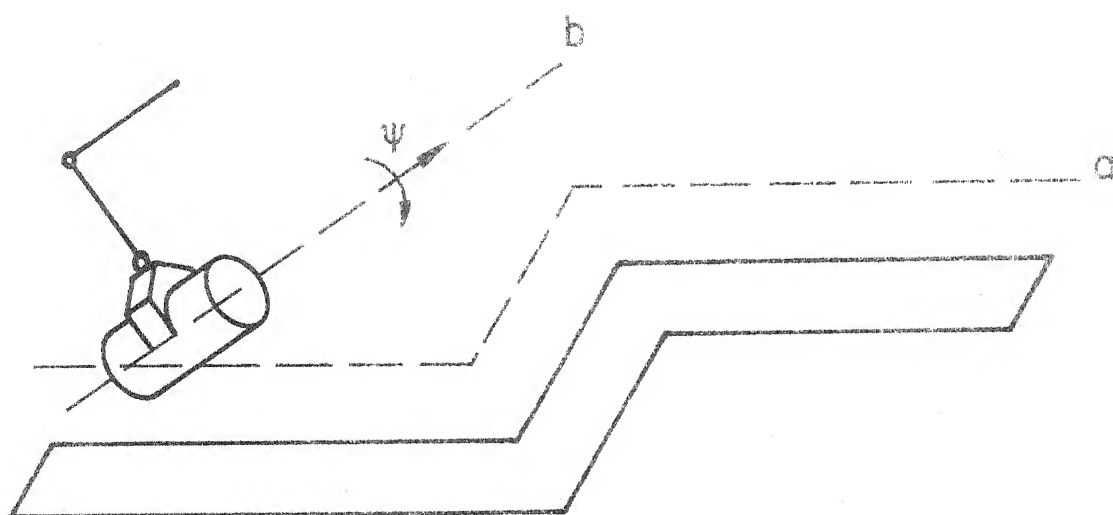
$$\dot{A} = \dot{A}_1 A_2 A_3 + A_1 \dot{A}_2 A_3 + A_1 A_2 \dot{A}_3$$

$$\ddot{A} = \ddot{A}_1 A_2 A_3 + \dot{A}_1 \dot{A}_2 A_3 + \dot{A}_1 A_2 \dot{A}_3 + \dots (3.12)$$

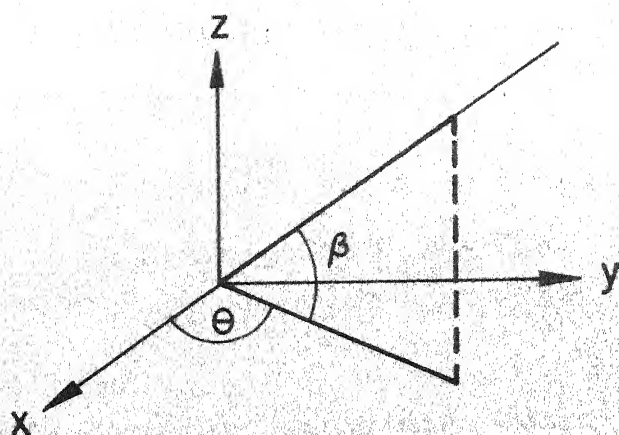
$$\dot{A}_1 \dot{A}_2 A_3 + A_1 \ddot{A}_2 A_3 + A_1 \dot{A}_2 \dot{A}_3 +$$

$$\dot{A}_1 A_2 \dot{A}_3 + A_1 \dot{A}_2 \dot{A}_3 + A_1 A_2 \ddot{A}_3$$

I.I.T. KANPUR  
CENTRAL LIBRARY  
87571  
No. A



(a) Spraying powder along a prescribed trajectory



(b) External angles  $\theta, \beta$

Fig. 3.5

B1 In this case, the manipulator has five dof, for which positioning is prescribed as in A1, A2 or A3 and the orientation is prescribed by the requirement that the EE should coincide with a given direction in space

$$\ddot{X}_F = [\ddot{x} \ \ddot{y} \ \ddot{z} \ \ddot{\theta} \ \beta]$$

where the angles  $\theta$  and  $\beta$  specify the direction, as in C2.  $w$  is calculated as in C2.  $\epsilon$  is calculated as follows.

$$\epsilon = \begin{bmatrix} \ddot{h}_1 \\ \ddot{h}_2 \end{bmatrix}$$

where

$$\begin{aligned} \ddot{h}_1 &= -\cos \beta \ \dot{\beta}^2 \cos \theta - \sin \beta \ \ddot{\beta} \cos \theta + \\ &\quad 2 \sin \beta \ \dot{\beta} \sin \theta \ \dot{\theta} - \cos \beta \cos \theta \ \dot{\theta}^2 \\ &\quad - \cos \beta \sin \theta \ \ddot{\theta} \\ \ddot{h}_2 &= -\cos \beta \ \dot{\beta}^2 \sin \theta - \sin \beta \ \ddot{\beta} \sin \theta - \\ &\quad - 2 \sin \beta \ \dot{\beta} \cos \theta \ \dot{\theta} - \cos \beta \sin \theta \ \dot{\theta}^2 \\ &\quad + \cos \beta \cos \theta \ \ddot{\theta} . \end{aligned} \quad \dots (3.13)$$

C3 In this case five functional coordinates are prescribed exactly as in B1 and the remaining dof is prescribed directly namely  $q_k(t)$

$$\ddot{X}_F = [\ddot{x} \ \ddot{y} \ \ddot{z} \ \ddot{\theta} \ \ddot{\beta} \ \ddot{q}_6]$$

The adjustment block is derived as in B1.



### 3.4 OTHER DYNAMICAL CHARACTERISTICS :

#### (a) P-N Diagram:

Since the drives and generalised velocities,  $q$ , are calculated at each step of the simulation, it is possible to draw a diagram for each joint, the characteristics of the driving motor torque  $P$  versus the motor rpm  $N$ , i.e. the diagram  $P$ - $N$  for each motor. Such a diagram is very useful during the synthesis and choice of the servosystems. The producer gives the  $P_{\max}$  -  $N$  motor characteristics in the catalogue where  $P_{\max}$  is the maximal motor torque at motor rpm  $N$ . By comparing the necessary characteristics obtained by means of simulation with the one from the catalogue, one can decide whether the chosen motor suits its applications.

The diagram  $P$ - $N$  is obtained in such a way that for each time instant and for joint  $k$ ,

$$N = R\dot{q}_k$$

$$P = \eta(R) P_k / N \quad \dots\dots (3.14)$$

where  $R$  is the reduction ratio of the subject joint and  $\eta(R)$  is its mechanical efficiency. Repeating the procedure for each time instant, the desired diagram is obtained.

## (b) Energy Consumed:

During the simulation, energy consumed during each step  $\Delta t_i$  is found, and the total energy is calculated by summation of these values.

$$E^i = E^{i-1} + E_{\Delta t_i} \quad \dots(3.15)$$

where  $E^i$  is the total energy consumed including during the  $i$ th time step, and  $E_{\Delta t_i}$  is the energy consumed during the  $i$ th time interval. To calculate  $E_{\Delta t_i}$  the average drive value on the interval is taken

$$P_{med}^i = \frac{1}{2} (P^{i-1} + P^i) \quad \dots(3.16)$$

Now,

$$E_{\Delta t_i} = \Delta q^{iT} \cdot P_{med}^i \quad \dots(3.17)$$

where  $\Delta q^i = q^i - q^{i-1}$ . The elements of the vectors  $q^i$  and  $P_{med}^i$  are absolute value. The average drive value is used to avoid more complex interpolation.

We can thus obtain the total energy consumed for the given manipulation task.

Thus, we now have a simulation algorithm which has the following inputs: the manipulator configuration, the manipulator state, and the manipulation task in the form most suited for prescribing ( according to the class of

functional movements). As for the output, we obtain time history of the generalised coordinates, velocities of the drives, the torque-rpm diagram for each actuator, and the total energy consumed.

## CHAPTER 1V

### THE PROGRAM MODULES

#### 4.1 INTRODUCTION:

The simulation algorithm described in the previous chapter is implemented on a DEC 1090 mainframe in an interactive, user friendly environment.

In one session, the user can set the input data, run the simulation routine and obtain the results on a VDU, a graphic terminal or the line printer, and save the output on the disk for future reference. If not satisfied with the results, the user may change any of the input parameters on line using the Editor routine, and rerun the Simulator routine. To effect the above actions is the Command Interpreter routine, which accepts commands from the user, decodes it, and performs the necessary action. The commands are arranged in a hierarchial fashion with three levels. Errors, if any, resulting either from user action or during the program run are trapped, and relevant warnings or messages given to the user.

#### 4.2 THE PROGRAM MODULES:

The program consist of four main modules. These will now be discussed in detail.

#### 4.2.1 Block Matrix Operator :

This module consists of a number of procedures which define all the vector, matrix and block matrix operations necessary for the simulation algorithm. Let us use the following notation to describe the vectors, matrices, and block matrices:

VEC-3D	A vector of dimension 1X3
VEC-ND	A vector of dimension 1XN
MAT-3D	A matrix of dimension 3X3
MAT-ND	A matrix of dimension NXN
BLK-V-V	A block vector of dimension 1XN, whose elements are of type VEC-3D
BLK-V-M	A block vector (1XN) with elements of type MAT-3D
BLK-M-V	A block matrix (NXN) with elements of type VEC-3D
BLK-M-M	A block matrix (MXN) with elements of types MAT-3D
BLK-V-V-V	A block matrix (1X2) with elements of type BLK-V-V
BLK-V-M-V	A block matrix (1X2) with elements of type BLK -M-V
BLK-M-M-M	A block matrix (2X2) with elements of type BLK-M-M.

An operation is named based on the type of the output variable. For instance, if a procedure takes two entities of type MAT-ND and after performing operations returns an entity of type MAT-ND, it will be named as OPN-MND. Similarly, if a procedure returns an entity of type BLK-M-M, it will be named as OPN-M-M. However, for operations involving dot product of vectors, the keyword DPN will be used instead of OPN. For describing the operation, the following notation will be used:

```
< Operation > (< entity,entity..>: <entity type>;
               < entity,entity..>: ...);
```

The following is the list of all the operations defined in this module. The last entity is always the output parameter.

```
i) OPN-V3D ( OPCODE: integer; S: real ; A: MAT-3D;
             B,C: VEC-3D; D: VEC-3D);
```

Here and in all further operations, OPCODE represents the operation code, whose value determines which operation is to be performed on the input parameters. In this case, the following operations are performed.

OPCODE	OPERATION
1	$D = B + C$
2	$D = B - C$
3	$D = S \cdot B$
4	$D = A \cdot B$

If the OPCODE is 1, then the vectors B and C are added according to the rules of the vector addition. Dummy variables may be passed for S and A. In this program, for dummy entities, a zero, or a null vector, or a null matrix, or a null block matrix is passed, depending upon the operations.

ii) OPN-VND ( OPCODE: integer; S:real; A,B,C: VEC-ND);

In Pascal, unlike Fortran, all the variables used must be declared, and the type checking is done at compile time. Hence, OPN-V3D is separated from OPN-VND, although the two differ only in the dimensions of the parameters. The OPCODE and the operations are exactly same as for i).

iii ) OPN-M3D ( OPCODE: integer; S: real; A,B,C;  
MAT-3D);

This is for doing operations on matrices. The Operations for the OPCODES are as follows

OPCODE	OPERATION
1	$C = A + B$
2	$C = A - B$
3	$C = A^T$
4	$C = S \cdot A$
5	$C = A \cdot B$

This procedure is used, for instance, to calculate the transition matrix  $A_N$  in Eqn. 2.6 by repeatedly applying it. Further, it is used, along with OPN-V3D and OPN-VND, in almost all the operation that follow.

iv) OPN - MND ( OPCODE : integer; S: real; A,B,C:  
MAT-ND);

This is same as OPN-M3D, except for the dimensions of the matrices A,B, and C, and the OPCODES have the same meaning. This procedure is called while evaluating the expressions in Eqn. 2.12 through 2.18. It is also called by some of the procedure that follow.

v) OPN1-V-V (OPCODE: integer; S:VEC-ND; A:BLK-M-V;  
B,C,D: BLK-V-V);

The operations performed by this procedure for different values of OPCODE is as follows



OPCODE	OPERATION
1	$D = B + C$
2	$D = B - C$
3	$D = S . A$

It is easy to see that while performing the operation  $D = B + C$ , the elements of B are added to the corresponding elements of C. These elements are not numbers, but vectors. Hence, to calculate D, this procedure invokes OPN-V3D, N times. Also, in this case, dummy variables are passed as S and A.

vi) OPN2-V-V (A:BLK-V-M; B:BLK-M-V; C:BLK-V-V);

No OPCODE is given here as only one operation is to be performed, i.e.,

$$C = A . B$$

An instance of the use of this operation is in Eqn. 2.19 for calculating  $D_i^1$  and  $D_i^2$ .

vii) OPN-M-V (OPCODE: integer; S: MAT-ND; A: BLK-M-M;  
B,C,D:BLK-M-V);

The OPCODE and corresponding operations are as follows;

OPCODE	OPERATION
1	$D = B + C$
2	$D = B - C$
3	$D = B^T$
4	$D = S . B$
5	$D = A . B$

This procedure calls OPN-V3D repeatedly to evaluate D. Also, when OPCODE is 3, the transpose of B is evaluated, not the elements of B. This procedure is used in Eqn. 2.15 through 2.19 to evaluate the expressions.

viii) OPN-M-M ( OPCODE: integer; S: MAT-ND; A,B,C:  
BLK-M-M) ;

This operates on block matrices whose elements are matrices, and is used for evaluating the block matrices B, D and C in Eqn. 2.13 through 2.18 and 2.21. The OPCODES and the corresponding operations are as follows:

OPCODE	OPERATIONS
1	$C = A + B$
2	$C = A - B$
3	$C = A^T$
4	$C = S \cdot A$
5	$C = A \cdot B$

Again, it should be noted that to evaluate C, the elements of A and the corresponding ones in B are operated by OPN-M3D repeatedly as the elements are not numbers but matrices.

ix ) OPN1-V-V-V (S:VEC-ND; A:BLK-V-M-V; B:BLK-V-V-V);

As there is only one operation to be performed, OPCODE is not used. This does the operation

$$B = A \cdot S$$

To evaluate expressions like  $B\dot{q}$  and  $D\dot{q}$  in Eqn. 2.12 and 2.16, this procedure is used.

x) OPN2-V-V-V (A,B,C: BLK-V-V-V);

This does the operation

$$C = A + B$$

Please note that as the elements of A and B are block matrices of type BLK-V-V, OPN-V-V is invoked repeatedly to evaluate  $C = A + B$ . This is used in Eqn. 2.16.

xi) OPN-V-M-V (A:BLK-V-M-V; B: BLK-M-M-M; C:BLK-V-M-V);

This is used in Eqn.2.23 and does the operation

$$G = B \cdot A$$

xii) DPN-R (A,B: VEC-3D; R: real);

This gives R as the dot product of two vectors A and B, and is used in the procedures that follow.

xiii) DPN-V (A: BLK-M-V;B:BLK-V-V; C:VEC-ND);

This performs the operation

$$C = B \cdot A$$

To do this, DPN-R is called repeatedly as the elements of B and A are vectors.

xiv) DPN1-M (A,B: BLK-M-V; C: MAT-ND) ;

This does the operation

$$C = A \cdot B$$

xv) DPN2-M (A,B: BLK-V-M-V; C:MAT-ND);

This does the operation

$$C = A \cdot B$$

by repeatedly applying DPN1-N to the elements of A and the corresponding ones of B and adding the matrices so obtained by OPN-MND.

There is also a procedure for calculating the inverse of a matrix.

xvi) INVERSE (DIM: integer; A,B: MAT-ND);

B is returned as the inverse of A. DIM is the dimension of the matrices A and B. This procedure is used to calculate the inverse of the Jacobian matrix, in Eqn. 3.5

#### 4.2.2 Simulator :

This module implements the simulation algorithm described in the previous chapter. This again consists of seven main submodules.

The first submodule creates the link structures for storing the input data and initialises certain variables and files. The configurations parameters of the links of the manipulator are represented internally by means of a doubly linked circular list, and are accessed by means of

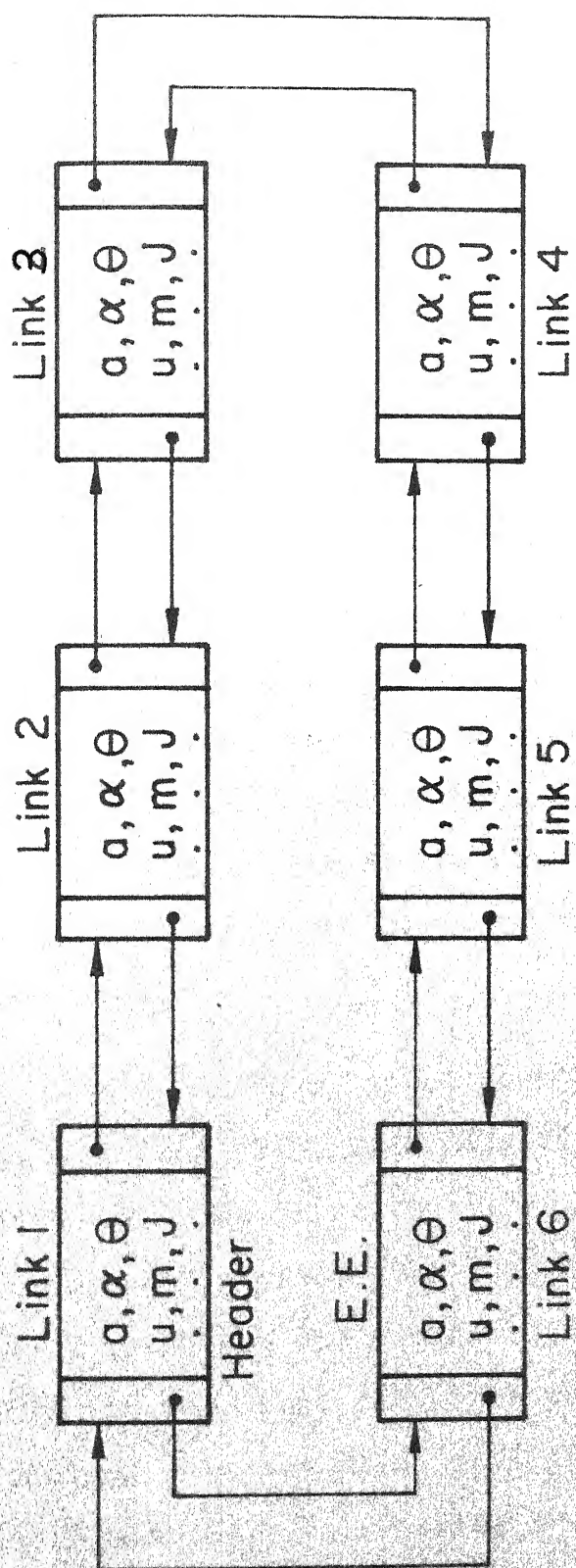


Fig.4.1 The link structure for representation of the manipulator configuration

a pointer, as shown in Fig. 4.1 . One node of the link is designated as the Header, and contains a record consisting of all the configuration parameters like linklength, link twist, mass, inertia, centre of gravity, etc. Also, the record contains two pointers, a left pointer pointing to the last link, and a right pointer pointing to the second link. In the same way, the left pointer of the second link contains the address of the pointer to the first link, and the right pointer, of the third. When the last link is reached its right pointer is linked to the first link. In the same manner, the generalised joint variable,  $q$ , velocity,  $\dot{q}$  and the transition matrix  $A$  are stored in another link structure. The acceleration, in functional coordinates, describing the task of the manipulator, is stored as an array ( for a series of time instants ) of an array ( for each dof ).

Four input channels are initialised for reading from files TASK.DAT, CONFIG.DAT, CMND.DAT and PROG.HLP. The first two files obviously are for task and configuration data, The sets of commands used by the Command Interpreter are stored in CMND.DAT, and PROG.HLP contains help for the user, which can be accessed while the program is running. The output files POSE.DAT, POSTR.DAT, DRIVE.DAT and PNDIAG.DAT, for storing the data for internal coordinates,

external coordinates, drives, and the P-N diagram respectively, are also initialised. Global variables like unit matrices, unit block matrices, matrices E and V in Eqn. 2.13 and 2.15, null matrices, etc. are initialised. Certain options are also set to their default values. The system of units for both input and output data are set to MKS units, the representation for positioning and orientation are set to Cartesian and Euler respectively ( Section 3.3 )

The second part of this module forms and solves the geometric and the kinematic model. First, the functions  $\lambda$  and  $\Lambda$  (Eqn. 2.10) are defined in procedures LAMDA and BIG-LAMDA. Another function, LAMDASTAR is defined, which takes two indices i and j, and returns the matrix  $\lambda(\rho_{i,j})$  (Eqn. 2.15). Next comes the procedure KINEMATIC. The geometric and kinematic modelling is done here. First the transition matrices are computed in the body fixed system, (Eqn. 2.3) and then the matrices so obtained are concatenated to obtain the transition matrices in the external coordinate system. Next the block matrix  $B^0$  is evaluated ( Eqn. 2.15). Now, the block matrix  $D^0$  is evaluated (Eqn. 2.19). Finally, internal velocities and accelerations are computed ( Eqn. 2.14, 2.18).

The third submodule computes the Jacobian matrix and the vector  $A$  from the matrices  $B^0$  and  $D^0$  evaluated earlier. ( Eqn. 3.4 ).

The fourth submodule is the adjustment block. The acceleration in functional coordinates are transformed into those in external coordinates, as discussed in Section 3.3.

Using the Jacobian matrix and the vector  $A$  obtained in submodule three, the accelerations in external coordinates obtained in submodule four are transformed into those in internal coordinates, in the submodule five.

The sixth submodule is for dynamic modelling.

Eqn. 2.20 through 2.25 are used and the drives, at a time  $t$ , obtained. Also, the torque and rpm are calculated for each joint at this time instant, as well as the energy consumed since the time instant  $( t - \Delta t )$ , from the Eqn. 3.14 and 3.17.

In the seventh submodule, the new state,  $q$  and  $\dot{q}$  are computed by the integration of  $\ddot{q}$  over the time interval  $\Delta t$ , and the time is incremented by  $\Delta t$ .

The first two main modules have now been discussed. But the user will really not be concerned with these two modules. To use the program, what he is interested in is the commands to be given, and the Editor.



#### 4.2.3 Command Interpreter (CI):

This is for performing simulation, editing, input/ output, etc., as and when requested by the user. The commands are organised in a hierarchial fashion into three levels. The commands can be shortened until it becomes unique. If the command is misspelt, or an illegal command or an ambiguous command is given at any level, a message is given to the user, warning this. If a valid command is given then it is interpreted, and the control passed to the module required to perform the action desired.

At the top level, a double arrow '>>' is given as a prompt . Now the user can type in any of the following commands:

Command	Action
>> HELP	Give help for top level actions
>> TYPE INPUT	Display the input data on the terminal
>> LIST INPUT	List the input data on the line pinter
>> SIMULATE	Perform simulation
>> TYPE OUTPUT	Display the output data on the terminal
>> LIST OUTPUT	List the output data on the line pinter

>> PLOT	Plot the graphs on the graphics terminal.
>> EDIT	Edit the input data.
>> UPDATE	Update the input file.
>> DOCUMENT	Type the Documentation.
>> EXIT	Exit from the top level.

When any of the input, output or edit action are requested by the user, the CI is invoked recursively, and the user enters the second level. For input action, the second level prompt is a vertical line '|'. Now the user can type in any of the following commands:

HELP	Type help for the input action commands.
SYSTEM	Type the system of units, coordinates, etc.
CONFIG	Type the configuration parameters of the manipulator.
STATE	Type the state parameters of the manipulator.
TASK	Type the task of the manipulator .
EXIT	Exit this level.

For output action, the second level prompt is a forward slash '/'. The user may now type in any of the following commands:

/ HELP	Type help for output action commands.
/ DRIVE	Display the drives in the joints.
/ ORIENT	Display the joint orientation .
/ POSITION	Display the joint positions .
/ VELOCITY	Display the joint velocities.
/ ACCELER	Display the joint accelerations.
/ PM DIAG	Display the P-M diagram.
/ ENERGY	Display the energy consumed.
/ EXIT	Exit this level.

For PLOT action, the second level prompt is a back slash '\ ' . The commands are same as for output actions.

For performing editing of the input data, when the EDIT command is given, then the user enters the second level. The prompt is a percent sign '%' . Now the following commands can be given :

% HELP	Type help for edit action.
% SYSTEM	Change system of units, representation of coordinates etc.
% CONFIG	Edit configuration parameters.
% STATE	Edit state parameters.
% TASK	Edit manipulator task.
% EXIT	End edit.

When one of the SYSTEM, CONFIG, STATE, or TASK command is given, the CI is once again called recursively, and the user thus enters the third level. The prompt is an asterix '\*'. If the user typed the SYSTEM command, the Editor asks for the system parameter to be changed. The user can now give the following answers:

- \* System Parameter : UNITS
- \* System Parameter : ANGLE
- \* System Parameter : POSITION
- \* System Parameter : ORIENT

The command interpreter is now once again invoked to determine the system of UNITS ( CGS, MKS or FPS), ANGLE (DEGREES or RADIANS), POSITION (CARTESIAN, CYLINDRICAL, or SPHERICAL), ORIENT ( NONE, PARTIAL, FULL). If the system of orientation is partial or full , CI is again called to determine whether the functional coordinates are Euler angles, a vector and an angle, etc ( as discussed in section 3.3 ). For instance,

- \* System of representation for Orientation: EULER.

In response to the CONFIG command, the editor gives the following prompt,

- \* Add node, delete node, or change node?

In response, the user can respond by typing ADD, DELETE, or CHANGE. If the CHANGE command is given, the editor asks for the number of the link whose link parameters are to be changed.

\* Link Number:

The user can now typing a number ( 1 to N ) for the link whose link parameters are to be changed . Now the C.I. is called to determine the particular parameter to be changed. The following commands can be given :

- \* Link parameter : LINKTYPE
- \* Link parameter : LINKLENGTH
- \* Link parameter : LINKTWIST
- \* Link parameter : JOINTDIST
- \* Link parameter : COG
- \* Link parameter : MASS
- \* Link parameter : INERTIA
- \* Link parameter : REDUCT
- \* Link parameter : EFF

After this, the user can type in the new values, and the corresponding changes will be made.

If the DELETE command is given, then the Editor asks for the number of the link to be deleted, and deletes that link. Similarly, for an ADD command, a number K is taken, and a new link added after the Kth link.

If the state parameters are to be edited, then the second level Editor command STATE can be given.

% STATE

Now the C.I. is invoked ( third level ) to determine the state parameter to be changed.

\* State Parameter:

Now the following commands may be given :

\* State parameter : INITIAL VELOCITY

\* State parameter : INITIAL POSIT

Now the vectors may be typed in as directed by the editor. Similarly, by issuing the second level % TASK command, the task parameters may be changed. The editor takes a number K from the user and changes the acceleration vector at the Kth time instant.

The user may make as many changes as he desires, then exit the editor module. Now the top level command SIMULATE may be given to rerun the simulation module. Now, the output parameters may be observed and evaluated. If the user is satisfied with the results, then he may issue the UPDATE command, which will update the input files CNFG.DAT and TASK.DAT. This cycle the user may continue as many times as he desires.

All the valid commands to the CI are stored in the file CMND.DAT as sets of commands. When the CI is called, a parameter is passed to it indicating which set of commands it should accept from the user. In a command set, for each command corresponds an action to be performed. The action for a particular command may include a call to the CI. By making such recursive calls to the CI, a hierarchical command structure is built. Control can pass to the higher level only after the lower level releases the CI. Such a structure allows new commands to be easily incorporated at any level, to cater to new modules which may be added to the program.

#### 4.2.4 Editor Module:

The editor may be called from the top level to interactively edit any of the input parameters. After editing the new input parameters may be saved by updating the file. All the Edit mode commands are described in Section 4.2.3.

Apart from the above four main modules the program also contains two other small modules: The HELP module for giving help to the user at various levels, and the UTILITY module which contains utility routines for performing certain actions which, though not directly related to the actual running of the program, help to make it more useable.

There are standard procedures for reading integers and reals from a device. Suppose the program is waiting for the user to give an real number from the terminal. If by mistake the user types an integer, or some character, the program is interrupted, and control returned to the monitor. To avoid this, two procedures, READIN, and READRL are written, which dont read the numbers directly, But as a sequence of characters which are converted into their ASCII code and then the number computed.

The whole program, except for the plotting routine is written in Pascal. The plotting routine, which calls certain routines from the Interactive, Graphics Library, PLOT-10, is written in Fortran, as these routines are only Fortran-Callable. The object code for the plotting routine and the rest of the program is linked together using the Fortran I/O Link Facility, FTNLNK, available here on DEC 10.

#### 4.3 SETTING INPUT DATA:

The data regarding the configuration, state, or task of the manipulator are initially given through disk files. These parameters may later be edited from the terminal.

The data for the configuration of the manipulator is given in the file CNFG . DAT in the following format:



NO OF LINKS

GRAVITY FORCE VECTOR

LINKTYPE/LINK LENGTH/LINK TWIST /JOINT ANGLE or  
JOINT DISTANCE

COG VECTOR

MASS

INERTIA TENSOR

EXTERNAL FORCE VECTOR

EXTERNAL MOMENT VECTOR

REDUCTION RATIO/ JOINT EFFICIENCY

A blank line or comment

LINKTYPE/LINKLENGTH /LINK TWIST/ JOINT ANGLE or  
JOINT DISTANCE

COG VECTOR

.

.

.

( for N links)

The LINKTYPE is given as 'R' for a revolute joint, and 'P' for a prismatic joint. The COG vector is given as  $(\rho_1 \rho_2 \rho_3)$ , and the inertia tensor J as  $(J_{11} J_{12} J_{13} J_{22} J_{23} J_{31})$ . In this way, the data for N links are given.

The data for the task of the manipulator, and the initial state are given in the file TASK.DAT in the following format :

INITIAL POSITION VECTOR

INITIAL VELOCITY VECTOR

A blank line or comment

FUNCTIONAL COORDINATES ACCELERATION

FUNCTIONAL COORDINATES ACCELERATION

.

.

.

FUNCTIONAL COORDINATES ACCELERATION

All these vectors are of the dimensions (  $1 \times N$  ) . The units for these values can be given in either MKS or CGS or FPS units. By default, MKS system of units is assumed.

#### 4.4 RUNNING THE PROGRAM:

To start the program, the following monitor command is given

.RUN ROBOTA

Now the control is passed to the top level CI and the flow of execution will now totally depend upon the commands given by the user.

## CHAPTER V

## RESULTS, CONCLUSIONS AND SUGGESTIONS

## 5.1 RESULTS :

The program is run on a DEC - 10 computer using a sample input data. The input and output data, together with the transactions in a single session are listed in Appendix A 1. The listing illustrates some of the facilities provided by the Editor. Also appendend with it are the drive time, state time, and torque-rpm graphs. The output data obtained agrees with the actual values computed manually. The program listing is given in Appendix A 2.

## 5.2 CONCLUSIONS:

As pointed out earlier, an algorithm which calculates the dynamic characteristics of a manipulator during the execution of a task clearly represents a useful means for the process of manipulator design. The algorithm chosen for this work forms and solves the dynamic model of the manipulator, and using incremental iterative displacement technique, computes the dynamic characteristics at a series of time instants. The dynamic model is based on the general theorem of Dynamics, and uses block matrix formalism to reduce the complex analytical equations into a compact form suitable for solving on a computer. The

inputs to the algorithm are the manipulator configuration (described by a set of parameters defining the geometric and inertial properties of the link), the initial state, and the manipulation task (in the form most suited for prescribing, according to the class of the functional movements). As the output, we obtain the time history of the generalised coordinates, velocities and drives, the P-N diagram, and the total energy consumed.

This algorithm is implemented in the increasingly popular Pascal language, to aid the designer in a fast analysis of the dynamic characteristics of a number of manipulator configuration. The P-N diagram can be used to give specifications for the actuator units. The total energy consumed for a particular task can be compared for different configurations and the configuration chosen accordingly. The various graphs can be viewed on a graphics terminal.

Any of the input parameters can be changed, by means of the Editor module and simulation done again. Further, the data may be given in any units. The program is command driven, i.e. the user types in commands for the action he desires. The command is decoded, and the action taken, by the Command Interpreter module. The Editor and

the Command Interpreter thus allow the user to analyse different configurations in an interactive manner. Further the structured development of the program allows new modules to be easily added.

### 5.3 SUGGESTIONS:

Improvements in the program can be made in two ways. One is an extension of the Dynamic module to include a manipulator with third and fourth class pairs, and a branched structure, and a manipulator with elastic segments. Secondly, other modules can be added to consider other problems in Robotics, like path synthesis, workspace synthesis actuator modelling, performance evaluation, etc. If this is done, coupled with the extensive use of computer graphics, then the gap between the designers and theoreticians can be bridged.

## REFERENCES

- [1] Denavit J. and Hartenberg R.S., 'A Kinematic Notation for Lower Pair Mechanisms based on Matrices', ASME Journal of Applied Mechanics, pp. 227-236, June 1955.
- [2] Hooker W.W. et Margulies G., 'The Dynamical Attitude Equations for a n-body satellite', The Journal of the Astronautical Sciences, Vol. XII, No.4, pp 123-128, Winter 1965.
- [3] Uicker J.J., 'Dynamic Behavior of Spatial Linkages', Transactions of the ASME, Journal of Applied Mechanics, pp.1-15, June 1969.
- [4] Wittenburg. J., 'Automatic construction of Non Linear Equations of Motion for Systems with many degrees of freedom', Euromech 38 Colloquim, Louvain La Neuve -Belgium, 3-5 Sept. 1973.
- [5] Zabala J., 'Automatic Computer Construction of equation of motion of Articulated Mechanisms', Proc. I Yugoslav Symp. on Industrial Robots, Beograd, Yugoslavia, 1977.
- [6] Vukobratovic, M., Potkonjak V., Scientific Fundamentals of Robotics 1: Dynamics of Manipulation Robots, Springer-Verlag, 1982.

- [7] Vukobratovic M., Stokic D., Scientific Fundamental of Robotics 2: Control of Manipulation Robots, Springer-Verlag, 1982.
- [8] Popov E., Modern Robot Engineering, Mir Publishers Moscow, 1982.
- [9] Fletcher H.J., Rongved L. et Yu E.Y., '' Dynamic Analysis of a Two-Body Gravitationally Oriented Satellite '', The Bell System Technical Journal, pp. 2239-2266, Sept. 1963.
- [10] Popov E.P., Vereschagin A.F., Zenkevich S.A., Manipulation Robots : Dynamics and Algorithms, '' Nauka '', 1978.

# APPENDIX A - 1

ROBBER TRANSACTION LOG  
 IIT KANPUR 5034(2)

10-04-83

7-MAY-85

02:35:33

Job 23 (13100,130461)

\*RUK 100K

02:35:101

\*C

>> STOP AT MAIN BEGIN  
 >> Type 4 for help  
 >>

\*C

< Continue , Skip or Quit > :

\*C

This is an interactive, user friendly, program, which can be used  
 for the design of Robotic Manipulators by the simulation of its dynam-  
 ics and other characteristics. The program was developed by Madhu Shekhar  
 under the guidance of Dr. B. Sahay.  
 For complete help see the DOCUMENTATION.  
 >>

\*HELP

The following is a summary of commands

```
-----
HELP          : type this text
TYPE INPUT    : type input parameters
LIST INPUT    : list input parameters
SIMULATE      : simulate manipulator dynamics
TYPE OUTPUT   : type output parameters
LIST OUTPUT   : list output parameters
PLOT          : plot graphs on graphics tty
EDIT          : edit input parameters
UPDATE        : update input file
DOCUMENT      : type the documentation
EXIT          : end execution
>>
```

\*TYPE U

>> Illegal command

\*TYPE INP

1

\*HELP

The commands are

```
-----
HELP          : type this text
SYSTEM        : display system of units, config, etc.
CONFIG        : display configuration parameters
STATE         : display manipulator state
TASK          : display manipulator task
EXIT          : end
```



# APPENDIX A - 1

OPSER TRANSACTION LOG  
 IIT KANPUR 503A(2)

10-04-83

7-MAY-83

02:35:35

Job 23 (13100,130461)

\*RU K 10 JK

[2:35:10]

\*C

>> STOP AT MAIN BEGIN  
 >> type h for help  
 >>

\*C

< Continue , Skip or Quit > :

\*C

This is an interactive, user friendly, program, which can be used  
 or the design of Robotic Manipulators by the simulation of its dyna-  
 nd other characteristics. The program was developed by Madhu Shekhar  
 under the guidance of Dr. S. Sahay.  
 For complete help see the DOCUMENTATION.  
 >>

\*HELP

The following is a summary of commands

```

-----
HELP          : type this text
TYPE INPUT    : type input parameters
LIST INPUT    : list input parameters
SIMULATE      : simulate manipulator dynamics
TYPE OUTPUT   : type output parameters
LIST OUTPUT   : list output parameters
PLOT          : plot graphs on graphics tty
EDIT          : edit input parameters
UPDATE        : update input file
DOCUMENT      : type the documentation
EXIT         : end execution
>>
  
```

\*TYPE U

>> Illegal command

\*TYPE INP

1

\*HELP

The commands are

```

-----
HELP          : type this text
SYSTEM        : display system of units, config, etc.
CONFIG        : display configuration parameters
STATE         : display manipulator state
TASK          : display manipulator task
EXIT         : end
  
```

```

0001 *SYSTEM
0002 Input data      : MKS
0003 Output data    : MKS
0004 Angle          : DEGREES
0005 Positioning of EE : CARTESIAN
0006 Orientation of EE : EULER
0007
0008
0009 *CONFIG
0010 This manipulator is an open chain active mechanism with 4 degrees
0011 of freedom
0012 < Continue , Skip or Quit > :
0013
0014 *C
0015 Link 1
0016 -----
0017 LINKTYPE      : Revolute
0018 LINKLENGTH    : 0.00000
0019 LINKTWIST     : 90.00000
0020 JOINTDIST     : 0.76500
0021 JOINTANGLE    : variable
0022 RHO           : 0.00000
0023             : -0.67499
0024             : 0.01499
0025             : 95.99000
0026 MASS         : 33.72399
0027 INERTIA       : 0.00000      0.00000      0.00000
0028             : 2.87900      0.60900      0.50900
0029             : 0.00000      0.60900      33.05500
0030 REDUCT. RATIO : 100.00000
0031 EFFICIENCY    : 0.85000
0032 < Continue , Skip or Quit > :
0033
0034 *C
0035 Link 2
0036 -----
0037 LINKTYPE      : Revolute
0038 LINKLENGTH    : 0.00000
0039 LINKTWIST     : -90.00000
0040 JOINTDIST     : 0.00000
0041 JOINTANGLE    : variable
0042 RHO           : 0.00000
0043             : 0.01000
0044             : -0.20300
0045 MASS         : 82.60999
0046 INERTIA       : 3.99000      0.00000      0.00099
0047             : 0.00000      3.78599      -1.53999
0048             : 0.00099      -1.53999      1.47499
0049 REDUCT. RATIO : 100.00000
0050 EFFICIENCY    : 0.85000
0051 < Continue , Skip or Quit > :
0052
0053 *C
0054 Link 3
0055 -----
0056 LINKTYPE      : Revolute
0057 LINKLENGTH    : -0.03500
0058 LINKTWIST     : 90.00000
0059 JOINTDIST     : 0.51000
0060 JOINTANGLE    : variable
0061 RHO           : 0.02900
0062             : -0.52400
0063             : -0.00700

```

```

0001 MASS : 52.900000
0002 INERTIA : 8.295000 0.17799 -0.01200
0003 : 0.17799 0.42500 0.19500
0004 : -0.01200 0.19500 8.23599
0005 REDUCT. RATIO : 100.00000
0006 EFFICIENCY : 0.85000
0007 < Continue , Skip or Quit > :

```

\*C

```

0008 Link 4
0009 -----
0010 LINKTYPE : Revolute
0011 LINKGEOMET : 0.00000
0012 LINKDIST : -90.00000
0013 JOINTDIST : 0.00000
0014 JOINTANGLE : Variable
0015 RAO : -0.00100
0016 : 0.02399
0017 : 0.00200
0018 MASS : 13.33999
0019 INERTIA : 0.15000 0.00000 -0.00099
0020 : 0.00000 0.06199 0.00099
0021 : -0.00099 0.00099 0.15000
0022 REDUCT. RATIO : 109.99999
0023 EFFICIENCY : 0.85000
0024 < Continue , Skip or Quit > :

```

\*C

\*STA

JOINT	JOINT ANGLE	JOINT VELOCITY
JOINT 1	29.9999	0.0000
JOINT 2	59.9999	0.0000
JOINT 3	-25.0000	0.0000
JOINT 4	10.0000	0.0000

\*TASK

TIME	JOINT1	JOINT2	JOINT3	JOINT4
0.00	0.421	-0.259	0.505	0.000
0.10	0.447	-0.255	0.508	0.000
0.20	0.490	-0.263	0.494	0.000
0.30	0.557	-0.266	0.486	0.000
0.40	0.624	-0.250	0.455	0.000
0.50	0.702	-0.223	0.414	0.000
0.60	0.842	-0.247	0.364	0.000
0.70	1.027	-0.127	0.383	0.000
0.80	1.146	-0.064	0.266	0.000
0.90	1.255	0.049	0.135	0.000
1.00	1.192	0.083	-0.098	47.555
1.09	1.155	-0.000	-0.295	47.555
1.19	1.192	0.108	-0.396	48.701
< Continue , Skip or Quit > :				

\*C

1.29	1.217	0.085	-0.483	48.701
1.39	1.329	0.153	-0.526	72.193
1.49	1.362	0.276	-0.536	74.484

0001	1.544	1.458	0.456	-0.530	74.484
0002	1.544	1.515	0.538	-0.424	75.630
0003	1.544	1.552	0.737	-0.278	64.744
0004	1.544	1.559	0.881	-0.171	57.869
0005	1.544	1.640	1.519	-0.315	54.431
0006	1.544	1.711	1.711	-0.377	48.701
0007	1.544	1.936	2.079	-0.292	53.285
0008	1.544	1.500	1.971	-0.344	51.566
0009	1.544	1.176	1.790	-0.373	52.712
0010	1.544	1.171	1.485	-0.641	0.000
0011	1.544	1.385	1.234	-0.515	0.000
0012	1.544	1.616	1.012	-0.571	0.000
0013	1.544	1.681	0.711	-0.602	0.000
0014	1.544	1.991	0.438	-0.705	0.000
0015	1.544	1.898	0.172	-0.530	36.669
0016	1.544	1.831	-0.169	-0.581	36.669
0017	> Continue , Skip or Quit > :				
0018	3.14	-0.430	-0.236	0.533	34.950
0019	3.14	-0.828	-0.284	0.507	35.523
0020	3.14	-0.474	-0.146	0.289	31.512
0021	3.14	-0.480	-0.123	0.299	31.512
0022	3.14	-0.426	-0.215	0.313	31.512
0023	3.14	-0.313	-0.140	0.251	0.000
0024	3.14	-0.304	-0.128	0.258	41.253
0025	3.14	-0.222	-0.076	0.223	41.253
0026	3.14	-0.036	-0.424	0.200	36.096
0027	3.14	-0.011	-0.473	0.198	37.242
0028	3.14	0.026	-0.535	0.495	0.000
0029	3.14	0.048	-0.603	0.194	0.000
0030	3.14	0.067	-0.569	0.151	0.000
0031	3.14	0.066	-0.533	0.127	0.000
0032	3.14	0.037	-0.476	0.130	0.000
0033	3.14	0.039	-0.472	0.114	6.875
0034	3.14	0.069	-0.335	0.124	63.025
0035	3.14	0.295	-0.106	-0.036	63.025
0036	*C				
0037	*O				
0038	*EXIT				
0039	>>				
0040	*E				
0041	>> Ambiguous command				
0042	>>				
0043	*SIM				
0044	>> Simulation going on. Please wait.				
0045	ENERGY CONSUMED = 95.573				
0046	>>				
0047	*TYPE OUT				



\*HELP

The commands are:

HELP : type this text  
DRIVE : display drives in joints  
POSITION : display joint positions  
VELOCITY : display joint velocity  
ACCELERATION : display joint accelerations  
% DIAG : display torque - Rpm diagram  
ENERGY : display energy consumption for task  
EXIT : end output parameters display

\*DEIV

/ illegal command

\*DRIV

< Continue , Skip or Quit > :

\*C

Five history of the drives in the joints

Time	Joint1	Joint2	Joint3	Joint4
0.000000	9.550660	3.641174	7.809998	11.797779
0.100000	10.195563	3.373355	8.165334	13.229339
0.200000	10.549009	3.592922	8.822334	14.135776
0.300000	11.510770	4.590331	10.559559	16.005888
0.400000	12.930661	5.840562	12.897229	18.802116
0.500000	11.206227	7.207770	15.362332	21.529778
0.600000	15.82987	8.95493	18.301338	24.806551
0.700000	21.82093	14.91414	26.11008	33.16501
0.800000	25.68384	17.03153	29.72158	39.68996
0.900000	29.12713	20.28629	40.33538	51.39901
1.000000	31.14123	25.13596	44.33157	56.33195
1.100000	25.95434	19.57750	35.10080	44.95842
1.199999	28.35542	24.11999	36.58231	43.56161

< Continue , Skip or Quit > :

\*C

1.299999	24.25456	19.21517	29.13220	37.16195
1.399999	23.59442	19.22044	28.42454	35.44696
1.499999	7.65959	3.68944	11.34444	18.53908
1.599999	7.57853	3.15075	12.29243	20.63906
1.699999	0.00000	-4.28888	2.01106	11.47763
1.799999	0.00000	-1.89722	-2.22520	6.24182
1.899999	0.00000	0.26757	-8.50038	-0.96956
1.999999	0.00000	5.55596	-14.86875	-13.68196
2.099999	-7.77058	-2.64225	-30.54229	-29.37927
2.199999	-4.84439	-0.44215	-25.69700	-23.63564
2.299999	-5.35187	-3.38068	-21.87200	-17.18837
2.399999	-3.64949	-2.22874	-16.04950	-11.94049
2.499999	-3.71529	-2.26640	-11.94905	-8.54431
2.599999	-4.34833	1.14603	-10.78065	-12.85998
2.699999	-6.02493	-0.18495	-9.68597	-12.55422
2.799999	-9.20560	-0.97751	-7.34047	-13.03264
2.899999	-11.47834	-2.73927	-6.53007	-12.99388

```

0001 2.99999 -15.75373 -2.45141 -4.16558 -15.68716
0002 3.09999 -17.53535 -3.97894 -9.54081 -21.70837
0003 < Continue , Skip or Quit > :

```

\*C

```

0004 3.19999 -11.73280 2.57790 -5.55620 -20.09090
0005 3.29999 -12.15486 2.26903 -5.47879 -20.27433
0006 3.39999 -12.55698 2.40968 -5.57997 -19.51913
0007 3.49999 -5.23023 1.71538 -0.77035 -7.89881
0008 3.59999 -5.01549 1.88652 -0.71155 -7.77182
0009 3.69999 0.00000 7.53746 -2.05727 -5.09085
0010 3.79999 0.00000 5.21749 0.91679 -3.47642
0011 3.89999 0.00000 5.14803 1.05359 -3.06064
0012 3.99999 0.00000 3.57904 1.52590 -0.73309
0013 4.09999 10.29451 12.48175 3.30452 1.78230
0014 4.19999 9.35372 12.02511 3.50478 2.58656
0015 4.29999 11.07205 13.19690 5.08054 4.59176
0016 4.39999 12.22592 14.35094 7.28044 6.41320
0017 4.49999 13.09303 14.81743 10.54045 10.40493
0018 4.59999 14.82292 16.12491 12.66084 13.21586
0019 4.69999 17.27947 18.53640 15.91539 16.71283
0020 4.79999 23.16658 24.06246 22.67859 24.09634
0021 4.89999 27.35844 28.15228 29.04551 30.58532
0022 5.00000 23.42359 26.42529 31.19131 35.99200
0023 < Continue , Skip or Quit > :

```

\*C

\*ACC

< Continue , Skip or Quit > :

\*C

Time history of the acceleration in the joints

```

-----
Time:      Joint 1      Joint 2      Joint 3      Joint 4
0.00000    0.90000    -1.00000    0.00000    0.00000
0.10000    0.95000    -1.12000    0.00000    0.00000
0.20000    0.95000    -1.14000    0.00000    0.00000
0.30000    0.97999    -1.14000    0.00000    0.00000
0.40000    1.02000    -1.17000    0.00000    0.00000
0.50000    1.02000    -1.17000    0.00000    0.00000
0.60000    1.02000    -1.17000    0.00000    0.00000
0.70000    1.25000    -1.17999    0.00000    0.00000
0.80000    1.29999    -1.45000    0.00000    0.00000
0.90000    1.29999    -1.50000    1.21000    0.00000
1.00000    1.34999    -1.51000    1.22999    0.00000
1.09999    0.92000    -1.15000    1.22999    0.82999
1.19999    0.92000    -0.70000    1.10999    0.82999
1.29999    0.73999    -0.70000    1.09999    0.85000
1.39999    0.73999    -0.70000    1.09999    0.85000
1.49999    0.73999    -0.70000    1.09999    0.85000
1.59999    0.73999    -0.70000    1.09999    0.85000
1.69999    0.73999    -0.70000    1.09999    0.85000
1.79999    0.73999    -0.70000    1.09999    0.85000
1.89999    0.73999    -0.70000    1.09999    0.85000
1.99999    0.73999    -0.70000    1.09999    0.85000
2.00000    0.73999    -0.70000    1.09999    0.85000

```

\*C

1.1.33333	0.730000	-0.450000	0.920000	0.850000
1.1.49999	0.230000	-0.340000	0.910000	1.260000
1.1.53333	0.240000	-0.330000	0.850000	1.299999
1.1.64999	0.330000	-0.330000	0.820000	1.299999
1.1.79999	0.000000	-0.100000	0.750000	1.320000
1.1.84999	0.000000	0.100000	0.800000	1.129999
1.1.99999	0.000000	0.910000	0.839999	1.010000
2.2.04999	0.750000	0.900000	0.000000	0.950000
2.2.14999	0.719999	0.860000	0.000000	0.850000
2.2.24999	1.250000	0.539999	0.000000	0.929999
2.2.39999	1.250000	0.539999	0.000000	0.900000
2.2.54999	1.179999	0.539999	0.000000	0.920000
2.2.69999	1.219999	1.070000	-0.519999	0.000000
2.2.79999	1.420000	1.070000	-0.650000	0.000000
2.2.84999	1.420000	1.310000	-0.650000	0.000000
2.2.99999	1.420000	1.310000	-0.650000	0.000000
3.3.04999	1.050000	1.820000	-0.650000	0.000000
3.3.14999	1.050000	1.820000	-1.349999	-0.640000
3.3.29999	1.020000	1.900000	-1.349999	-0.640000

\*C

3.3.29999	-1.020000	1.900000	-1.349999	-0.610000
3.3.39999	-1.050000	1.900000	-1.250000	-0.619999
3.3.49999	-0.519999	1.099999	-0.450000	-0.550000
3.3.59999	-0.519999	1.099999	-0.450000	-0.550000
3.3.69999	0.000000	1.099999	-0.300000	-0.550000
3.3.79999	0.000000	0.810000	-0.300000	0.000000
3.3.84999	0.000000	0.810000	-0.300000	0.719999
3.3.99999	0.000000	0.619999	-0.150000	0.719999
4.4.04999	1.150000	0.510000	-0.150000	0.630000
4.4.19999	1.150000	0.500000	-0.150000	0.650000
4.4.29999	1.299999	0.500000	0.000000	0.000000
4.4.39999	1.400000	0.500000	0.000000	0.000000
4.4.49999	1.400000	0.450000	0.619999	0.000000
4.4.59999	1.450000	0.400000	0.630000	0.000000
4.4.69999	1.489999	0.400000	0.580000	0.000000
4.4.79999	1.719999	0.350000	0.500000	-0.120000
4.4.84999	1.719999	0.350000	0.500000	-1.099999
4.4.99999	1.500000	0.000000	0.500000	-1.099999

\*C

/ Illegal command

\*VEL

< Continue , Skip or Quit > :

\*C

-----Time history of the velocities in the joints-----

Time	Joint 1	Joint 2	Joint 3	Joint 4
0.00000	0.00000	-0.10000	0.00000	0.00000



0.100000	0.135000	-0.212000	0.000000	0.000000
0.200000	0.280000	-0.326000	0.000000	0.000000
0.300000	0.378000	-0.440000	0.000000	0.000000
0.400000	0.420000	-0.557000	0.000000	0.000000
0.500000	0.582000	-0.674000	0.000000	0.000000
0.600000	0.684000	-0.791000	0.000000	0.000000
0.700000	0.809000	-0.909000	0.000000	0.000000
0.800000	0.933000	-1.034000	0.000000	0.000000
0.900000	1.059000	-1.204000	0.121000	0.000000
1.000000	1.204000	-1.355000	0.24399	0.000000
1.100000	1.295000	-1.470000	0.36699	0.08299
1.200000	1.338000	-1.540000	0.47800	0.16599
1.300000	1.452000	-1.610000	0.58799	0.25099

< Continue , Skip or Quit > :

\*C

1.399999	1.532000	-1.665000	0.680000	0.33599
1.499999	1.555000	-1.689000	0.77099	0.46200
1.599999	1.579000	-1.722000	0.85599	0.59200
1.699999	1.579000	-1.755000	0.93799	0.72200
1.799999	1.579000	-1.785000	1.01299	0.85400
1.899999	1.579000	-1.755000	1.09299	0.96699
1.999999	1.579000	-1.664000	1.17700	1.06800
2.099999	1.504000	-1.558000	1.17700	1.16300
2.199999	1.432000	-1.482000	1.17700	1.24799
2.299999	1.307000	-1.429000	1.17700	1.34099
2.399999	1.182000	-1.374000	1.17700	1.43099
2.499999	1.057000	-1.320000	1.17700	1.52300
2.599999	0.939000	-1.213000	1.12500	1.52300
2.699999	0.817000	-1.106000	1.06000	1.52300
2.799999	0.675000	-0.975000	0.99500	1.52300
2.899999	0.533000	-0.844000	0.93000	1.52300
2.999999	0.368000	-0.662000	0.86500	1.52300
3.099999	0.203000	-0.480000	0.73000	1.45900
3.199999	0.101000	-0.290000	0.59500	1.39500

< Continue , Skip or Quit > :

\*C

3.299999	-0.000000	-0.100000	0.460000	1.33400
3.399999	-0.100000	0.089999	0.335000	1.27199
3.499999	-0.152999	0.199999	0.290000	1.21700
3.599999	-0.204999	0.309999	0.245000	1.16200
3.699999	-0.204999	0.419999	0.215000	1.10700
3.799999	-0.204999	0.500000	0.185000	1.10700
3.899999	-0.204999	0.581999	0.155000	1.17900
3.999999	-0.201999	0.643999	0.140000	1.25100
4.099999	-0.089999	0.694999	0.125000	1.31400
4.199999	0.025000	0.744999	0.110000	1.37900
4.299999	0.155000	0.794999	0.110000	1.37900
4.399999	0.295000	0.844999	0.110000	1.37900
4.499999	0.436000	0.889999	0.172000	1.37900
4.599999	0.581000	0.929999	0.235000	1.37900
4.699999	0.730000	0.969999	0.293000	1.37900
4.799999	0.902000	1.004999	0.343000	1.36700
4.899999	1.074000	1.039999	0.393000	1.25700



5.00000 1.22100 1.03999 0.44300 1.14700  
/ illegal command  
/

\*PR

< Continue , Skip or Quit > :

\*C

The PD Diagram

JOINT 1

RP4

TORQUE

85.94000	0.08118
178.55519	0.08666
267.37199	0.08966
350.93219	0.09784
458.33200	0.10991
555.75183	0.12075
653.15158	0.13455
772.51409	0.18547
896.55110	0.21831
1020.73810	0.24758
1119.89959	0.29020

< Continue , Skip or Quit > :

\*C

1237.55040	0.22061
1325.40121	0.24110
1396.05380	0.20616
1462.90680	0.20055
1484.85950	0.06519
1507.78713	0.06441
1507.78713	0.00000
1507.78713	0.00000
1507.78713	0.00000
1507.78713	0.00000
1507.78713	0.00000
1436.15952	-0.06604
1367.41682	-0.04117
1248.05434	-0.04549
1128.69183	-0.03272
1009.32933	-0.03158
896.55113	-0.03696
780.15333	-0.05121
644.55753	-0.07824
508.95173	-0.09756

< Continue , Skip or Quit > :

\*C



```

001  -1.4159403  -0.00375
002  -1.3939403  -0.02873
003  -1.3829403  -0.01894
004  -1.2909403  -0.01926
005  -1.1989403  -0.00974
006  -1.0969403  -0.00156
007  -0.9949403  -0.00830
008  -0.8929403  -0.02322
009  -0.7909403  -0.02053
010  -0.6889403  -0.03382
011  < 0.5869403  , Skip or Quit > :

```

```

012  1273.932104  0.02191
013  1355.450000  0.01928
014  35.940000  0.02048
015  160.373000  0.01483
016  296.018000  0.01603
017  401.057000  0.05406
018  478.404000  0.04434
019  555.751000  0.04375
020  611.455000  0.03042
021  664.555000  0.06009
022  711.450000  0.10221
023  759.115000  0.11217
024  806.890000  0.12198
025  849.860000  0.12594
026  886.000000  0.13706
027  923.252000  0.15755
028  959.674000  0.20453
029  993.095000  0.23937
030  993.095000  0.22461
031  < Continue , Skip or Quit > :

```

```

032  < Continue , Skip or Quit > :

```

### JOINT 3

RPM	TORQUE
0.00000	0.06638
0.00000	0.05940
0.00000	0.07498
0.00000	0.08975
0.00000	0.10962
0.00000	0.13057
0.00000	0.15555
0.00000	0.22193
0.00000	0.25263
115.54290	0.34285
292.99560	0.37681
350.44830	0.29835
456.44221	0.31094

\*C < Continue , Skip or Quit > :

561.45121	0.24762
649.33202	0.23160
736.22791	0.09642
817.39442	0.10148
895.59523	0.01709
957.31371	-0.01891
1043.70573	-0.07225
1123.91732	-0.12638
1123.91732	-0.25960
1123.91732	-0.21842
1123.91732	-0.18591
1123.91732	-0.13642
1123.91732	-0.10156
1071.26252	-0.09163
1012.19102	-0.08233
950.12552	-0.06239
889.05752	-0.05550
825.98851	-0.03540
697.07752	-0.08109

\*C < Continue , Skip or Quit > :

563.15552	-0.05572
439.25102	-0.05506
319.39152	-0.04742
276.92102	-0.00654
233.95052	-0.00604
205.30351	0.01748
176.55552	0.00779
148.00952	0.00904
133.68802	0.01296
119.36252	0.02808
105.03902	0.03234
105.03902	0.04828
105.03902	0.06188
164.24282	0.08959
224.40152	0.10761
279.78572	0.13528
327.53073	0.19276
375.27573	0.24688
423.02073	0.26512

\* < Continue , Skip or Quit > :

\*C

\*C < Continue , Skip or Quit > :

\*C

JOINT 4  
-----

RPM	TORQUE
0.00000	0.09116

)

0.00000	0.10222
0.00000	0.10923
0.00000	0.12358
0.00000	0.14528
0.00000	0.15836
0.00000	0.19158
0.00000	0.23627
0.00000	0.30669
0.00000	0.39717
0.00000	0.43529
0.00000	0.34740
87.14235	0.33661
171.30473	

< continue , skip or quit > :

\*C

263.54756	0.28716
352.93403	0.27390
485.28017	0.14325
521.33067	0.15948
758.33157	0.08869
897.33305	0.04823
1015.72711	-0.60719
1121.81651	-0.10572
1221.60355	-0.22702
1310.88569	-0.18263
1408.57295	-0.13281
1503.10403	-0.09226
1599.74396	-0.06602
1599.74396	-0.09937
1599.74396	-0.09700
1599.74396	-0.10070
1599.74396	-0.10040
1599.74396	-0.12121
1532.51899	-0.16774

< continue , skip or quit > :

\*C

1465.29403	-0.15524
1401.22024	-0.15666
1335.09607	-0.15082
1278.32463	-0.05103
1220.55313	-0.06005
1162.78174	-0.03933
1162.78174	-0.02686
1238.40983	-0.02365
1314.03788	-0.00566
1380.21244	0.01377
1448.48781	0.01998
1448.48781	0.03548
1448.48781	0.04955
1448.48781	0.08040
1448.48781	0.10212
1448.48781	0.12914
1435.88311	0.18619
1320.34023	0.23634
1204.79734	0.27812

\*C < Continue , Skip or Quit > :

\*C < Continue , Skip or Quit > :

\*HELP

The commands are:

```
HELP      :: type this text
DRIVE     :: display drives in joints
POSITION  :: display joint positions
VELOCITY  :: display joint velocity
ACCELERATION :: display joint accelerations
PM DIAG   :: display Torque - Rpm diagram
ENERGY    :: display energy consumption for task
EXIT      :: end output parameters display
```

\*ENERGY

\*C < Continue , Skip or Quit > :

Time history of the ENERGY consumption

\*C < Continue , Skip or Quit > :

Time Cumulative Energy

0.000000	0.01238
0.100000	0.09342
0.200000	0.24087
0.300000	0.44705
0.400000	0.71132
0.500000	1.03025
0.600000	1.38893
0.700000	1.77981
0.800000	2.28804
0.900000	3.14488
1.000000	4.60680
1.099999	6.62840
1.199999	9.05036

\*C < Continue , Skip or Quit > :

1.299999	11.97972
1.399999	15.31400
1.499999	18.33119
1.599999	20.93636
1.699999	23.33017
1.799999	24.56225

```

0001 1.000000 25.22356
0002 1.000000 25.30659
0003 1.000000 25.38638
0004 1.000000 25.44630
0005 1.000000 27.26556
0006 1.000000 28.01795
0007 1.000000 28.60922
0008 1.000000 29.40439
0009 1.000000 30.62130
0010 1.000000 32.32364
0011 1.000000 34.17930
0012 1.000000 36.03450
0013 1.000000 38.70742
0014 < Continue , Skip or Quit > :

```

```

0015 3.000000 41.40761
0016 3.000000 43.83021
0017 3.000000 46.24371
0018 3.000000 47.90289
0019 3.000000 48.50909
0020 3.000000 49.17960
0021 3.000000 49.33037
0022 3.000000 49.40648
0023 3.000000 49.46260
0024 4.000000 50.17543
0025 4.000000 51.42637
0026 4.000000 53.03684
0027 4.000000 55.26048
0028 4.000000 58.27361
0029 4.000000 62.25598
0030 4.000000 67.39534
0031 4.000000 74.56405
0032 4.000000 84.26932
0033 5.000000 95.57339
0034 < Continue , Skip or Quit > :

```

```

0035 < Continue , Skip or Quit > :

```

-----  
 Fine history of the internal coordinates of the joints  
 -----

Time	Joint 1	Joint 2	Joint 3	Joint 4
0.00000	0.52809	1.04219	-0.43632	0.17453
0.10000	0.54184	1.02659	-0.43632	0.17453
0.20000	0.56509	0.99969	-0.43632	0.17453
0.30000	0.59799	0.96139	-0.43632	0.17453
0.40000	0.64089	0.91154	-0.43632	0.17453
0.50000	0.69399	0.84999	-0.43632	0.17453
0.60000	0.75729	0.77674	-0.43632	0.17453



001  
002  
003  
004  
005  
006  
007  
008  
009  
010  
011  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
054  
055  
056  
057  
058  
059  
060

0.70000	0.83194	0.69174	-0.43632	0.17453
0.30000	0.41934	0.59359	-0.43632	0.17453
0.90000	1.01974	0.48069	-0.43027	0.17453
1.00000	1.13339	0.35274	-0.41202	0.17453
1.09999	1.25839	0.21149	-0.38147	0.17866
1.19999	1.39259	0.06099	-0.33922	0.19113
1.29999	1.53559	-0.09650	-0.28592	0.21198

< Continue , Skip or Quit > :

1.34399	1.58479	-0.25975	-0.22252	0.24133
1.14999	1.43911	-0.42695	-0.14997	0.28123
1.53399	1.99531	-0.59750	-0.06862	0.33393
1.53399	2.15374	-0.77135	-0.02107	0.39963
1.79999	2.31164	-0.94735	0.11862	0.47843
1.89999	2.45954	-1.12335	0.22392	0.56948
1.99999	2.62744	-1.29430	0.33742	0.67123
2.09999	2.78159	-1.45590	0.45512	0.78278
2.19999	2.92839	-1.60840	0.57282	0.90333
2.29999	3.05534	-1.75390	0.69052	1.03278
2.39999	3.18979	-1.89940	0.80822	1.17138
2.49999	3.30174	-2.02870	0.92592	1.31908
2.59999	3.40154	-2.15535	1.04102	1.47138
2.69999	3.48934	-2.27130	1.15027	1.62368
2.79999	3.56394	-2.37535	1.25302	1.77598
2.89999	3.62434	-2.46630	1.34927	1.92628
2.99999	3.66939	-2.54160	1.43902	2.08058
3.09999	3.69794	-2.59870	1.51877	2.22968
3.19999	3.71314	-2.63720	1.58502	2.37238

< Continue , Skip or Quit > :

3.29999	3.71814	-2.65670	1.63777	2.50883
3.39999	3.71304	-2.65720	1.67752	2.63913
3.49999	3.70034	-2.64270	1.70877	2.76358
3.59999	3.68244	-2.61720	1.73552	2.88253
3.69999	3.65194	-2.58070	1.75852	2.99598
3.79999	3.64144	-2.53465	1.77852	3.10668
3.89999	3.62094	-2.48050	1.79552	3.22098
3.99999	3.60044	-2.41920	1.81027	3.34248
4.09999	3.58559	-2.35225	1.82352	3.47073
4.19999	3.58244	-2.28025	1.83527	3.60538
4.29999	3.59144	-2.20325	1.84527	3.74328
4.39999	3.61394	-2.12125	1.85727	3.88118
4.49999	3.65049	-2.03450	1.87137	4.01908
4.59999	3.70134	-1.94350	1.89172	4.15698
4.69999	3.76689	-1.84850	1.91812	4.29488
4.80000	3.84849	-1.74975	1.94992	4.43218
4.90000	3.94729	-1.64750	1.98672	4.56338
5.00000	4.06219	-1.54350	2.02852	4.68358

/

/ Illegal command

\*C

\*EXIT



```

001
002
003
004
005
006 >>
007 *PLOT
008
009 * Please go to the Graphics room
010 >>
011 *ED
012
013
014 *HELP
015
016 The following is a summary of commands
017 -----
018 HELP : type this text
019 SYSTEM : change the system of units, coordinates
020 etc.
021 CONFIG : change the manipulator configuration
022 STATE : change the manipulator state
023 TASK : change the manipulator task
024 EXIT : end edit
025
026 *CONFIG
027
028 * Add, delete or change :
029 *CHANGE
030
031 * linkno :
032
033 *1
034
035 * Configuration parameter :
036
037 *LINKT
038
039 * Configuration parameter : Ambiguous command
040 * Configuration parameter :
041
042 *LINKTWISI
043
044 * value :
045
046 *75.0
047
048 * Configuration parameter :
049
050 *EXIT
051
052 * Add, delete or change :
053
054 *CHA
055
056 * linkno :
057
058 *3
059
060 * Configuration parameter :
061
062 *LINKTW
063
064 * value :
065
066 *75.0
067
068 * Configuration parameter :
069
070 *EX

```

```

001
002
003
004
005
006 * Add,delete or change :
007 *CH
008
009 * linkno :
010 *4
011
012 * Configuration parameter :
013 *PHD
014
015 * value :
016 * value : Rho11 :
017 *-0.00095
018
019 * value : Rho12 :
020 *0.025
021
022 * value : Rho13 :
023 *0.00205
024
025 * Configuration parameter :
026 *EXIT
027
028 * Add,delete or change :
029 *EXIT
030
031 %
032 *CONFIG
033
034 * Add,delete or change :
035 *C
036
037 * linkno :
038 *4
039
040 * Configuration parameter :
041 *MASS
042
043 * value :
044 *16.5
045
046 * Configuration parameter :
047 *EXIT
048
049 * Add,delete or change :
050 *EXIT
051
052 %
053 *EXIT
054
055 >>
056 *TYPE INP
057
058
059
060

```

\*C34

This manipulator is an open chain active mechanism with 4 degrees  
of freedom  
< Continue , Skip or Quit > :

\*C

Link 1  
-----  
LINKTYPE : Revolute  
LINKLENGTH : 0.00000  
LINKTWIST : 75.00000  
JOINTDIST : 0.70500  
JOINTANGLE : variable  
RHO : 0.00000  
-0.67499  
0.01499  
MASS : 95.99000  
INERTIA : 33.72399 0.00000 0.00000  
0.00000 2.87900 0.60900  
0.00000 0.00000 33.05500  
REDUCT. RATIO : 100.00000  
EFFICIENCY : 0.85000  
< Continue , Skip or Quit > :

\*C

Link 2  
-----  
LINKTYPE : Revolute  
LINKLENGTH : 0.00000  
LINKTWIST : -90.00000  
JOINTDIST : 0.00000  
JOINTANGLE : variable  
RHO : 0.00000  
0.01000  
-0.20300  
MASS : 82.60999  
INERTIA : 3.99000 0.00000 0.00099  
0.00000 3.78599 -1.53999  
0.00099 -1.53999 1.47499  
REDUCT. RATIO : 100.00000  
EFFICIENCY : 0.85000  
< Continue , Skip or Quit > :

\*C

Link 3  
-----  
LINKTYPE : Revolute  
LINKLENGTH : -0.03500  
LINKTWIST : 75.00000  
JOINTDIST : 0.51000  
JOINTANGLE : variable  
RHO : 0.02900  
-0.52400  
-0.00700

```

001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060

```

MASS : 52.90000  
 INERTIA : 8.29500 0.17799 -0.01200  
 : 0.17799 0.42500 0.19500  
 : -0.01200 0.19500 8.23599  
 REDUCT. RATIO : 100.00000  
 EFFICIENCY : 0.85000  
 < Continue , Skip or Quit > :

\*C

```

015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060

```

DISC 1  
 DISCTYPE : Revolute  
 DISKMSGT1 : 0.00000  
 DISKINIST : -90.00000  
 JOINTDIST : 0.00000  
 JOINTANGHE : Variable  
 RAD : 0.00095  
 : 0.02500  
 : 0.00205  
 MASS : 16.50000  
 INERTIA : 0.15000 0.00000 -0.00099  
 : 0.00000 0.05199 0.00099  
 : -0.00099 0.00099 0.15000  
 REDUCT. RATIO : 100.99999  
 EFFICIENCY : 0.85000  
 < Continue , Skip or Quit > :

\*C

```

033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060

```

I EXIT  
 \*SIMULATE  
 >> Simulation going on. Please wait.  
 ENERGY CONSUMED = 121.06260  
 >>

\*TYPE OUT

\*DR

< Continue , Skip or Quit > :

\*C

Time history of the DRIVES in the joints

Time	Joint1	Joint2	Joint3	Joint4
0.00000	13.74070	6.86060	11.78751	9.01055
0.10000	14.61155	6.78617	12.43711	10.39404
0.20000	14.94331	6.96051	13.12725	11.42567
0.30000	15.00524	7.96021	14.89157	13.28165
0.40000	17.55056	9.21791	17.30698	16.08421
0.50000	18.74850	10.40795	19.68270	18.97062
0.60000	23.27353	11.94359	22.45527	22.31926
0.70000	27.12037	18.31072	30.25411	29.17317
0.80000	31.01602	20.47518	34.35925	35.83124
0.90000	34.25520	23.41013	41.16369	42.32123
1.00000	39.23971	28.03960	44.63923	45.72986

1.09999	29.27542	20.75691	33.80249	35.18084
1.19999	31.55845	24.83170	35.46454	32.54878

< Continue , Skip or Quit > :

1.29999	26.75618	19.90562	29.51930	26.92664
1.39999	25.94739	20.29283	30.64755	25.05974
1.49999	8.15862	4.75859	13.52935	9.40936
1.59999	3.44115	5.39502	15.22059	9.48382
1.69999	0.00000	-1.41716	3.35324	-1.62262
1.79999	0.00000	-2.10216	-2.18844	-9.47385
1.89999	0.00000	5.35024	-10.15875	-19.26290
1.99999	0.00000	11.80573	-17.95385	-33.38384
2.09999	-11.87157	1.29276	-44.19419	-54.01735
2.19999	-3.77884	3.11530	-40.81881	-47.85647
2.29999	-12.10316	-2.21180	-41.10342	-41.07900
2.39999	-11.00895	-1.88985	-34.50279	-32.45561
2.49999	-10.88441	-2.64212	-28.95513	-25.53740
2.59999	-11.05545	-0.21481	-27.25913	-27.30132
2.69999	-12.84924	-2.11287	-24.26042	-24.07183
2.79999	-15.99039	-4.02481	-20.72690	-22.72671
2.89999	-19.09877	-6.06154	-17.57055	-20.54141
2.99999	-24.43332	-6.96184	-14.60843	-22.94565
3.09999	-29.23392	-8.65044	-13.95872	-28.02934

< Continue , Skip or Quit > :

3.19999	-15.95342	-1.27479	-13.87507	-24.60248
3.29999	-17.35402	-1.65030	-13.47501	-24.87649
3.39999	-17.14415	-1.51727	-12.32801	-24.22572
3.49999	-3.84579	-0.42354	-4.96890	-10.65112
3.59999	-8.67497	-0.27490	-4.99085	-10.62833
3.69999	0.00000	6.24599	-1.01713	-6.79545
3.79999	0.00000	4.31932	-2.27541	-4.81206
3.89999	0.00000	4.28553	-2.13139	-4.43416
3.99999	0.00000	2.98343	-1.12453	-1.83719
4.09999	15.39353	14.33736	2.06039	3.14982
4.19999	15.01261	13.97583	3.02046	4.01671
4.29999	19.00766	15.61871	5.68716	6.26371
4.39999	19.67315	17.23994	7.75531	7.94189
4.49999	20.53692	18.13171	12.38553	11.78267
4.59999	22.37938	20.04864	14.62174	13.61500
4.69999	24.89546	23.17763	17.56753	15.56604
4.80000	31.72333	30.03334	24.03828	21.10753
4.90000	35.61730	35.12112	29.75720	25.04080
5.00000	35.28149	33.72117	30.95708	27.54543

< Continue , Skip or Quit > :

/

< Continue , Skip or Quit > :

The PN diagram

JOINT 1

001  
002  
003  
004  
005  
006  
007  
008  
009  
010  
011  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
054  
055  
056  
057  
058  
059  
060

RP4

TORQUE

85.94049	0.11579
176.65549	0.12419
267.37199	0.12701
358.96219	0.13004
454.35200	0.14917
555.75130	0.15936
653.15124	0.17232
772.51409	0.23052
896.55110	0.26363
1020.78413	0.29116
1144.54954	0.33353

< Continue , Skip or Quit > :

\*C

1237.55010	0.24865
1325.40121	0.26825
1396.05345	0.22742
1462.90040	0.22055
1484.86950	0.07189
1507.74713	0.07175
1507.78713	0.00000
1507.78713	0.00000
1507.78713	0.00000
1507.78713	0.00000
1436.16962	-0.09920
1367.11632	-0.07462
1245.05434	-0.10542
1128.69183	-0.09357
1009.32933	-0.09251
896.55113	-0.09397
780.15333	-0.10921
644.55753	-0.14441
508.95173	-0.16233

< Continue , Skip or Quit > :

\*C

351.40324	-0.20768
193.84474	-0.22298
96.44494	-0.14410
-0.95485	-0.14750
-96.44485	-0.14572
-146.09965	-0.07518
-195.75445	-0.07373
-195.75445	0.00000
-195.75445	0.00000
-195.75445	0.00000
-195.75445	0.00000
-85.94095	0.13934
23.87254	0.13610
148.00954	0.15306
281.69554	0.16722
416.33844	0.17456
554.79693	0.19022
697.07702	0.21161
861.31982	0.25964

< Continue , Skip or Quit > :



```

001
002
003
004
005 *C
006 1023.55264 0.30274
007 1163.70764 0.29989
008 < Continue , Skip or Quit > :
009
010 *Q
011 /
012 *EE
013 < Continue , Skip or Quit > :
014
015 *C
016 time history of the ENERGY consumption
017 -----
018
019 Time Cumulative Energy
020 0.00000 0.01376
021 0.10000 0.10224
022 0.20000 0.26092
023 0.30000 0.48429
024 0.40000 0.77590
025 0.50000 1.13565
026 0.60000 1.55207
027 0.70000 2.03524
028 0.80000 2.67239
029 0.90000 3.70013
030 1.00000 5.36794
031 1.09999 7.57005
032 < Continue , Skip or Quit > :
033
034 *C
035 1.19999 10.10647
036 1.29999 13.09022
037 1.39999 16.42408
038 1.49999 19.28614
039 1.59999 21.42387
040 1.69999 22.83583
041 1.79999 23.15594
042 1.89999 24.45757
043 1.99999 27.00784
044 2.09999 28.38400
045 2.19999 30.68638
046 2.29999 33.00632
047 2.39999 35.39698
048 2.49999 37.47598
049 2.59999 39.54023
050 2.69999 41.82240
051 2.79999 44.50722
052 2.89999 47.50759
053 2.99999 50.79153
054 < Continue , Skip or Quit > :
055
056 *C
057 3.09999 54.42280
058 3.19999 57.60974
059 3.29999 60.37838
060 3.39999 63.15331
061 3.49999 65.20423
062 3.59999 66.45513
063
064
065 3.69999 67.32159
066 3.79999 67.75973
067 3.89999 68.09264
068 3.99999 68.27485
069 4.09999 69.06593
070 4.19999 70.65021
071 4.29999 72.69942
072 4.39999 75.52394
073 4.49999 79.29503
074 4.59999 84.14935
075 4.69999 90.18897
076 4.80000 98.30549
077 4.90000 108.98030
078 < Continue , Skip or Quit > :
079
080 *C
081 5.00000 121.06260
082 /
083 *HELP
084 The commands are:
085 -----
086 DRIVE : type this text
087 POSITION : display drives in joints
088 VELOCITY : display joint positions
089 : display joint velocity

```

```

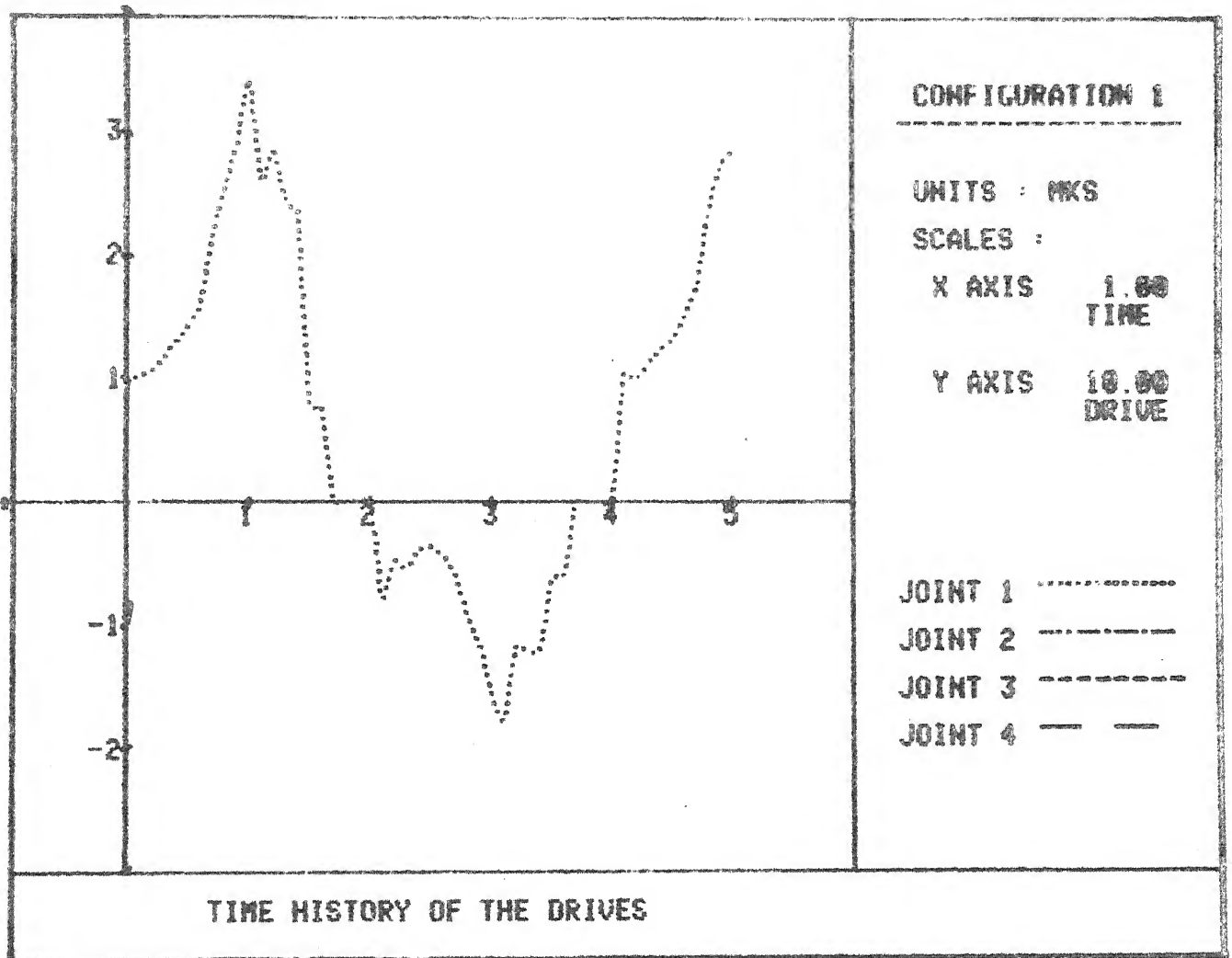
090 PM DIAG      : Display Torque - Rom diagram
091 ENERGY      : display energy consumption for task
092 EXIT         : end output parameters display
093 /
094
095 *EX           >>
096
097 *DOC
098
099 *INTRO
100
101 dynamic performance is one of the most significant
102 actors in designing mechanical arms particularly for fast
103 and accurate robots recently developed.
104 Until now the choice of the kinematic scheme and its
105 different parameters , actuator and the control systems
106 hits was a subject of free speculation, frequently based on
107 experience but lacking any systematic method. Hence the need
108 or developing certain criteria and procedures for a
109 systematic choice of manipulator design. In the design
110 phase, there is no efficient means except simulation to
111 investigate and evaluate the highly non linear and coupled
112 systems. The aim of my work was to create a software tool
113 or the simulation of all the dynamical values and
114 characteristics of manipulator operations in a particular
115 task execution and thus permit a fast evaluation of a great
116 number of different configurations.
117 < Continue , Skip or Quit > :
118
119 *QUIT
120
121 *EXIT
122

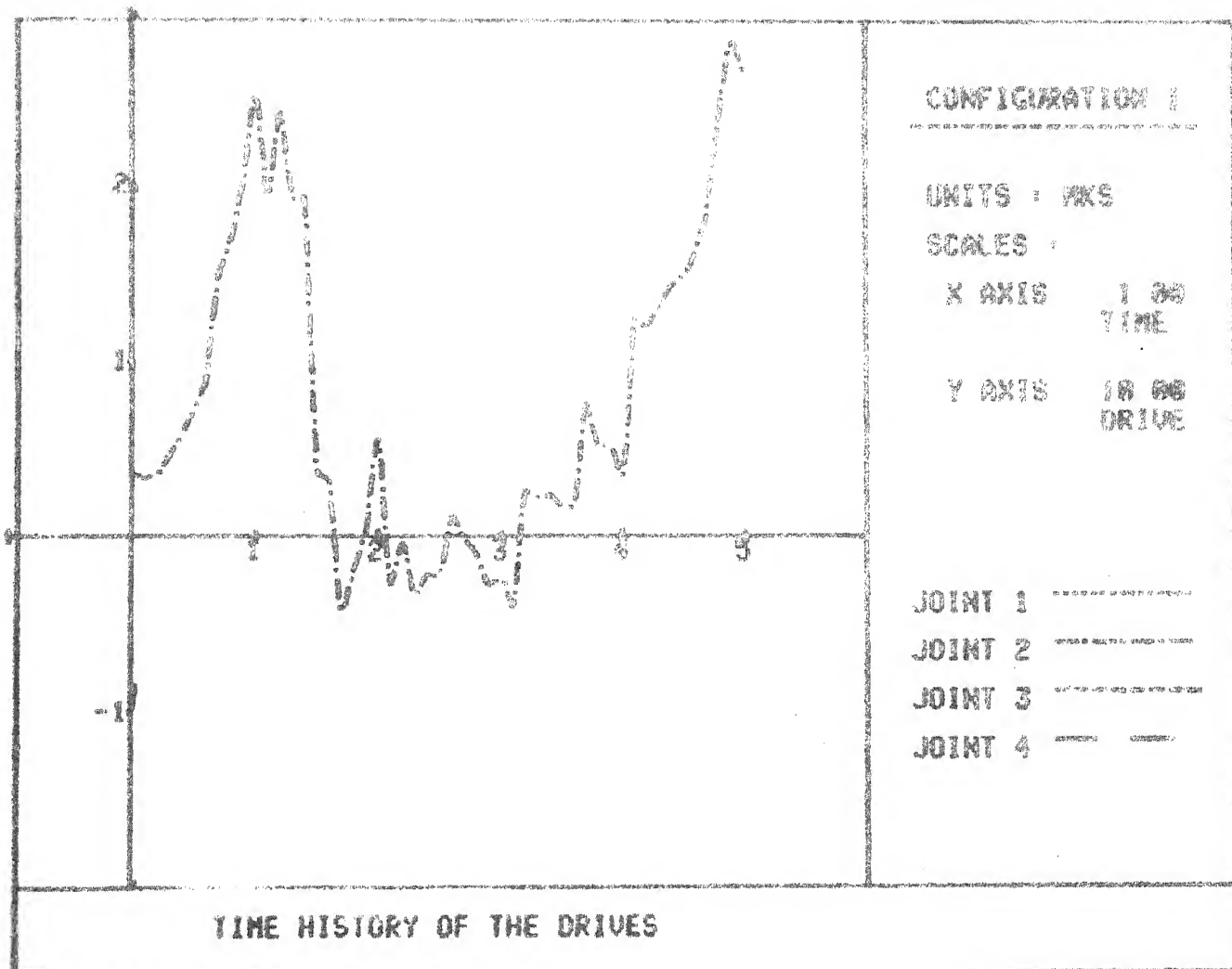
```

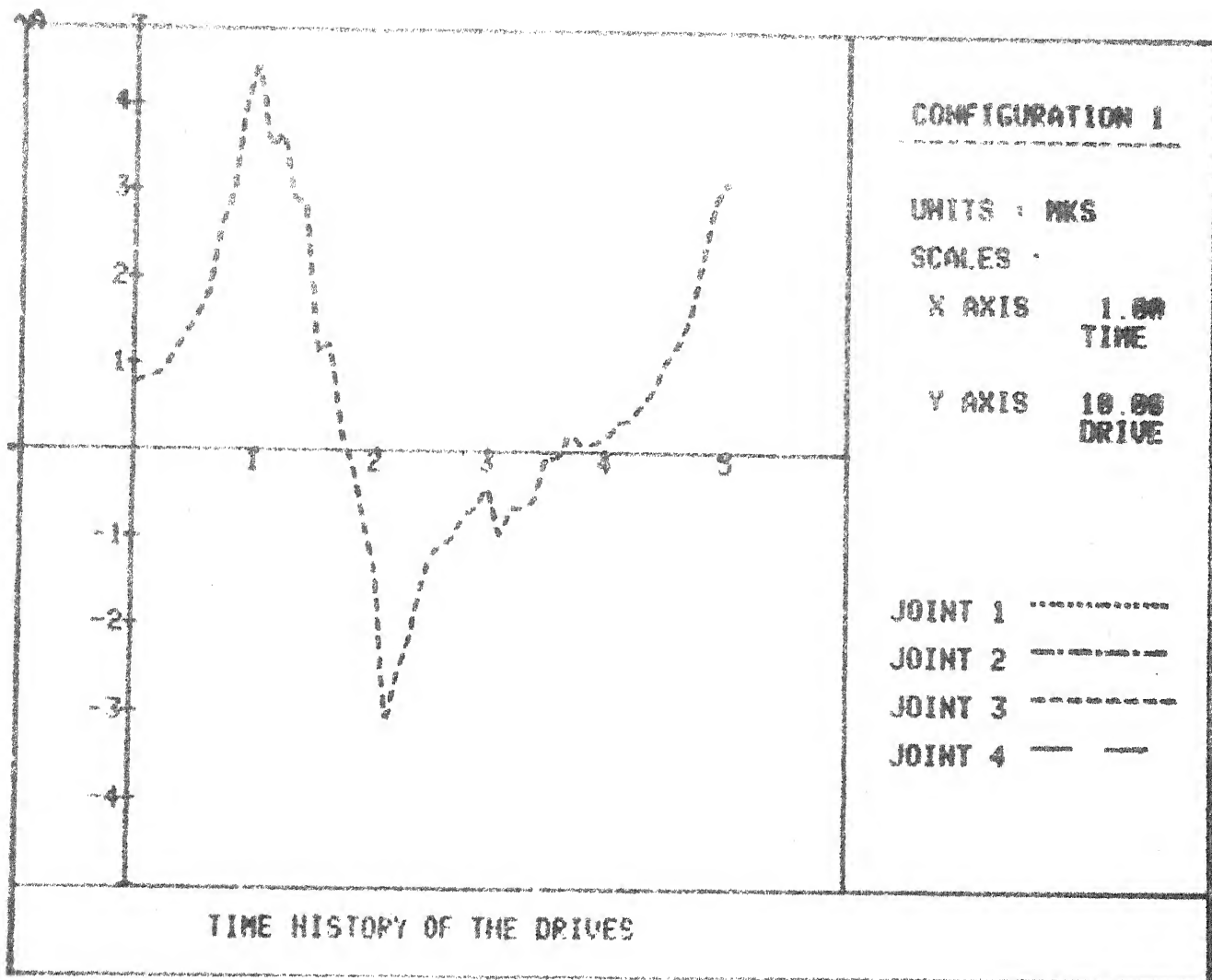


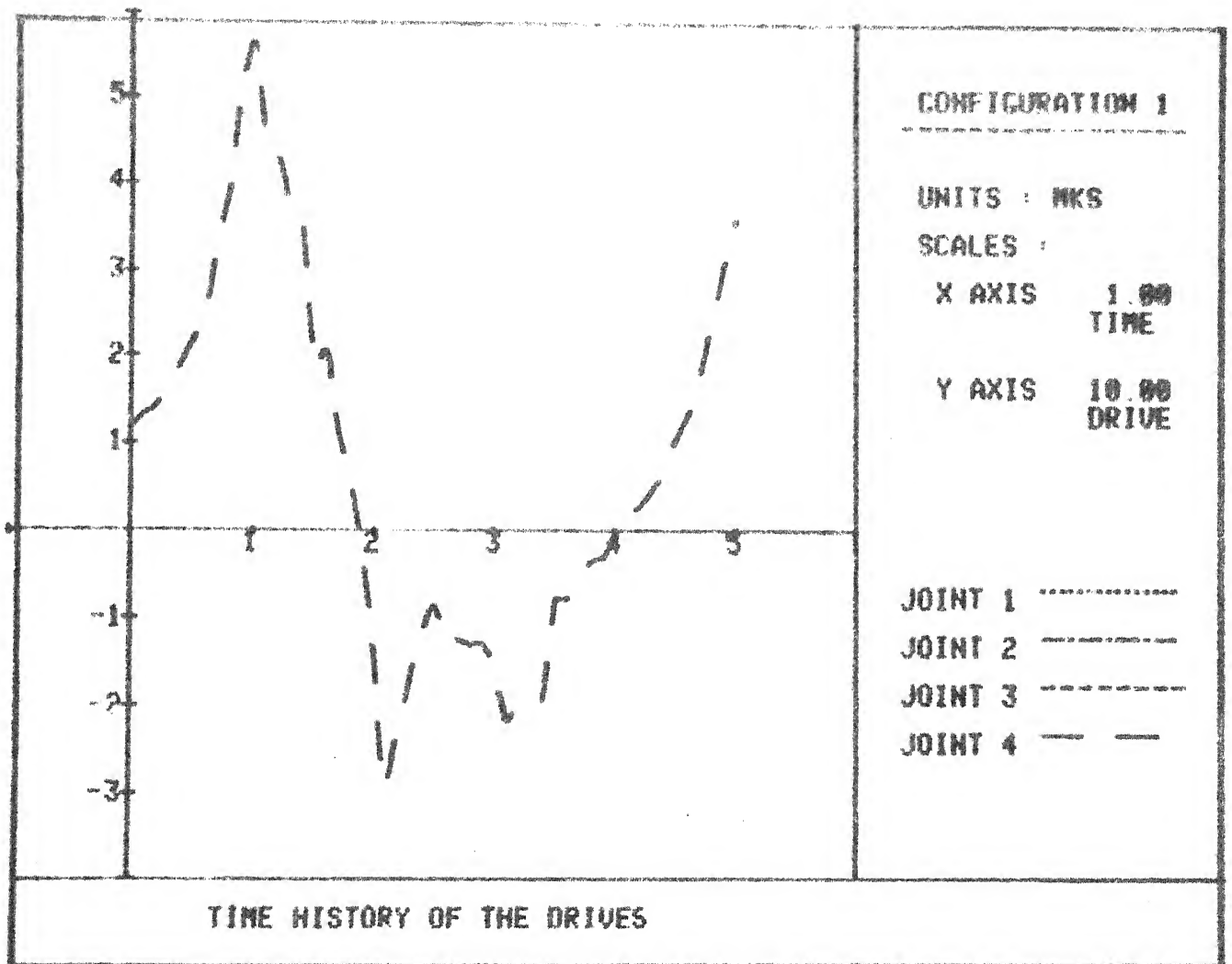
001  
002  
003  
004  
005  
006

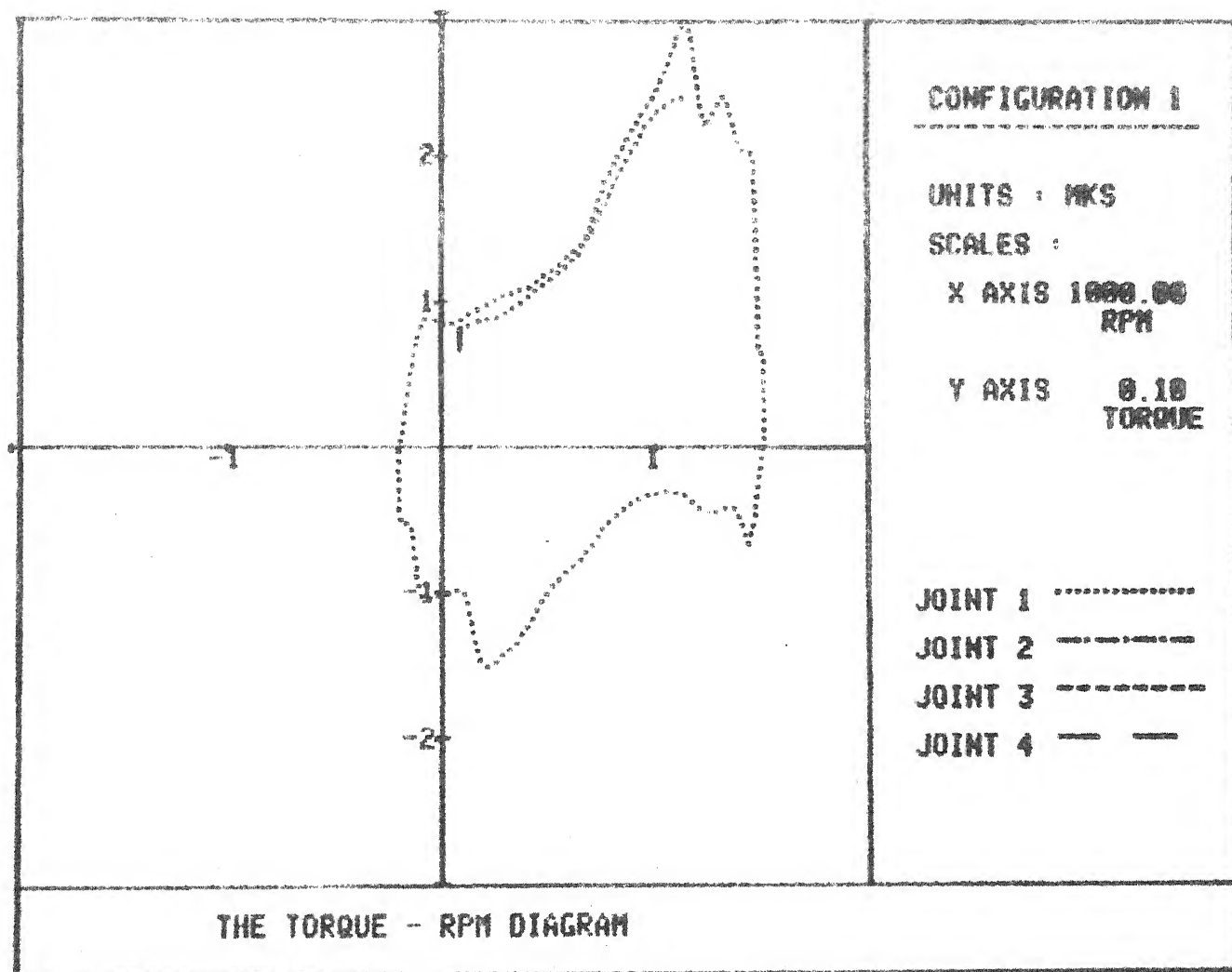
\*EXIT

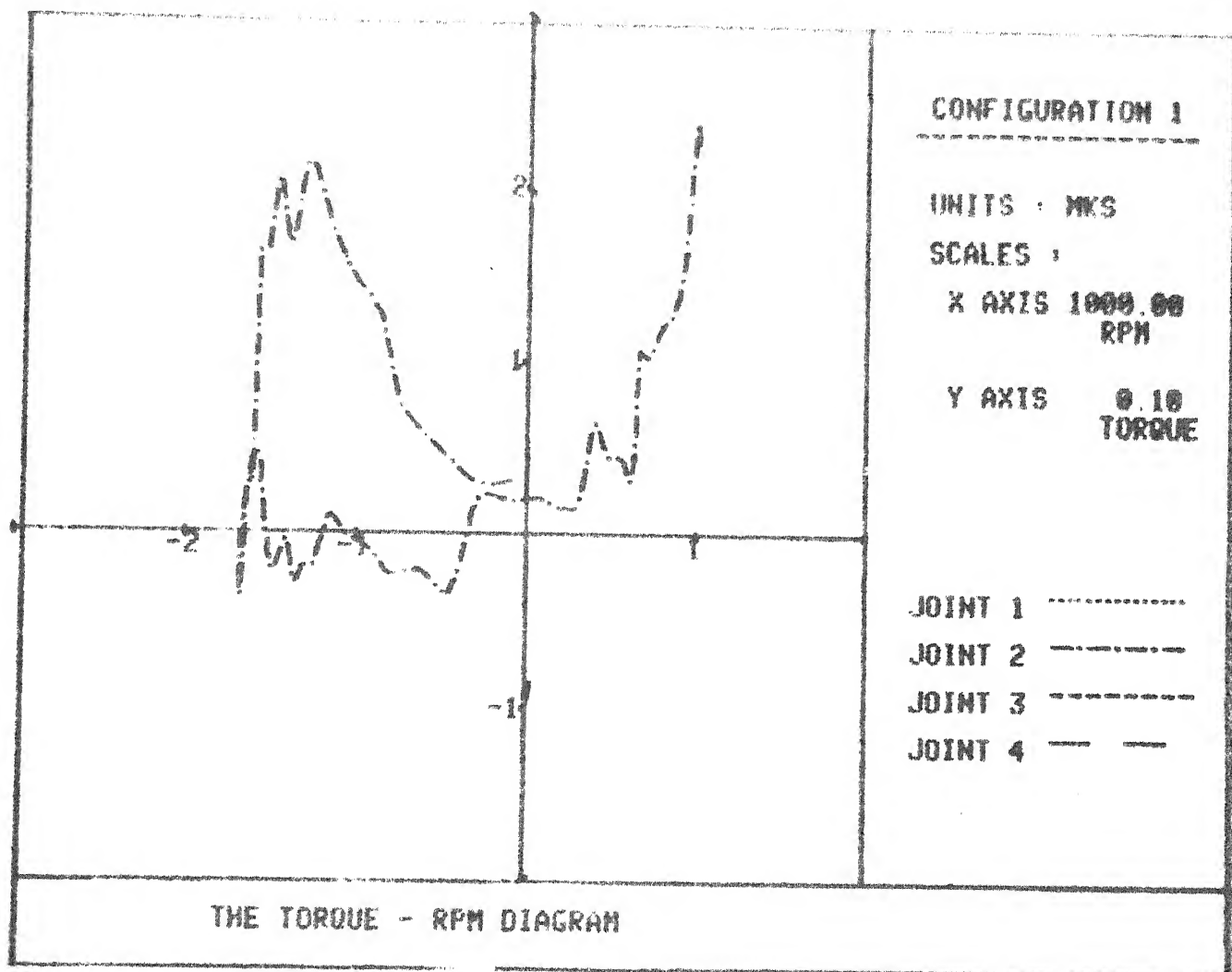


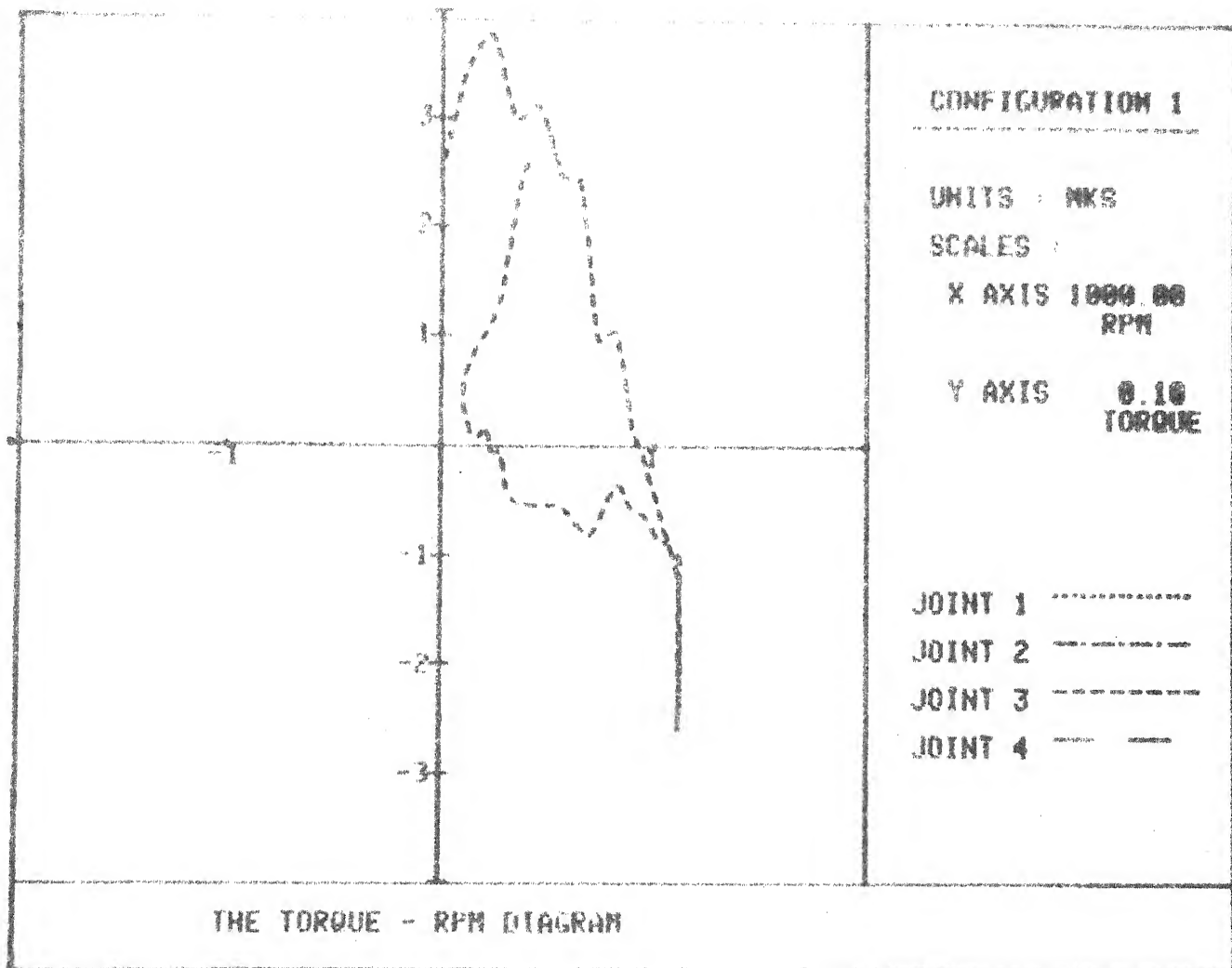




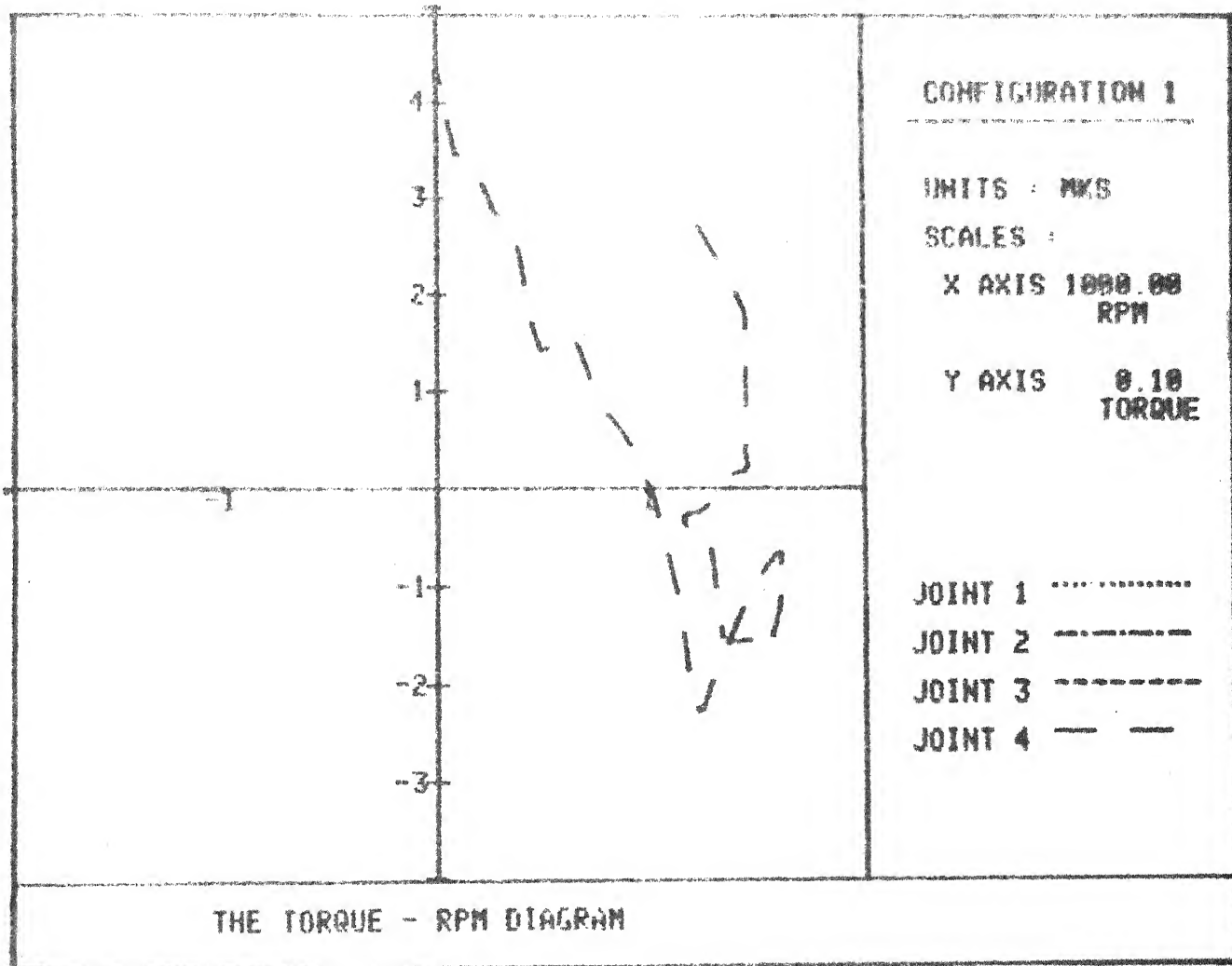


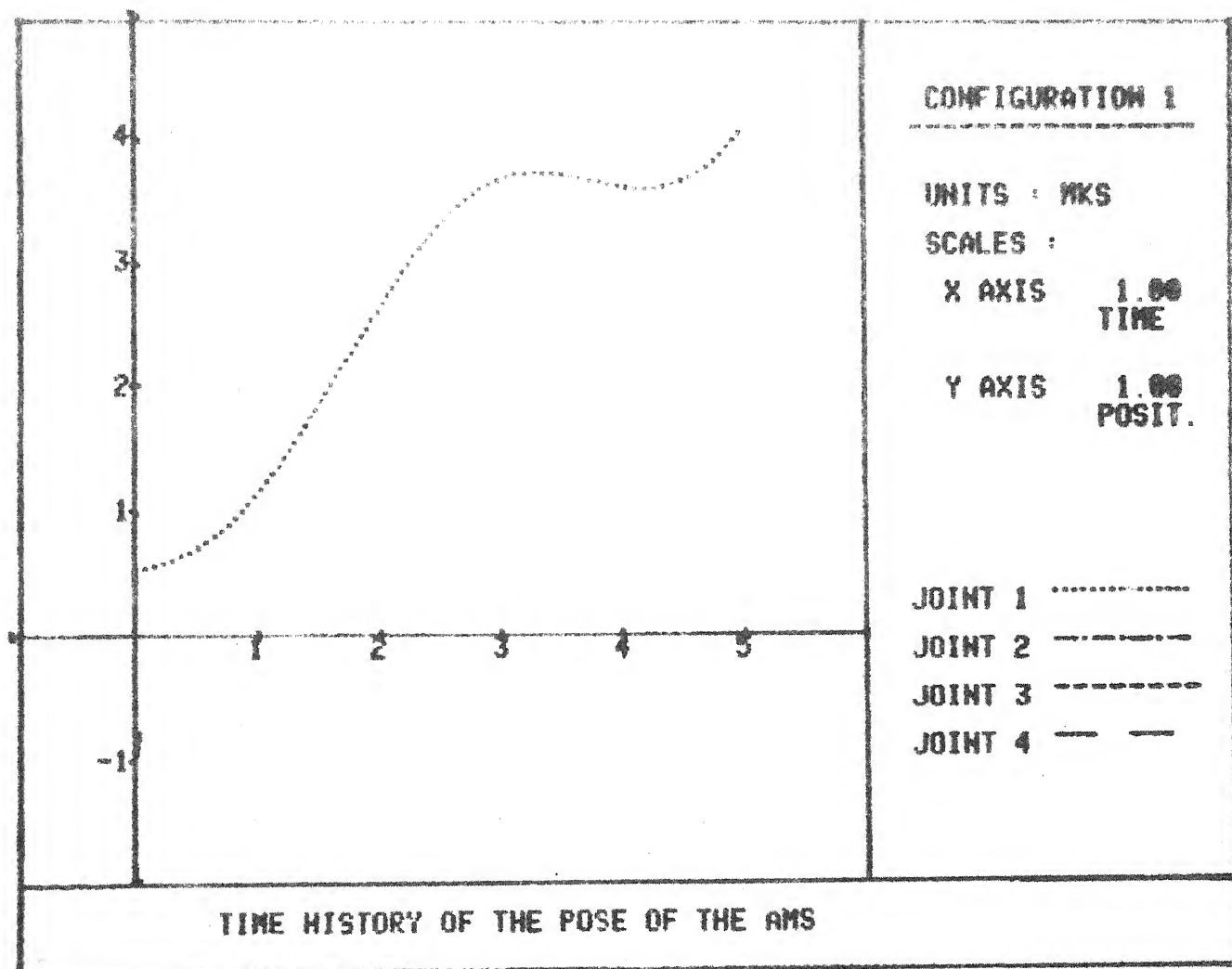


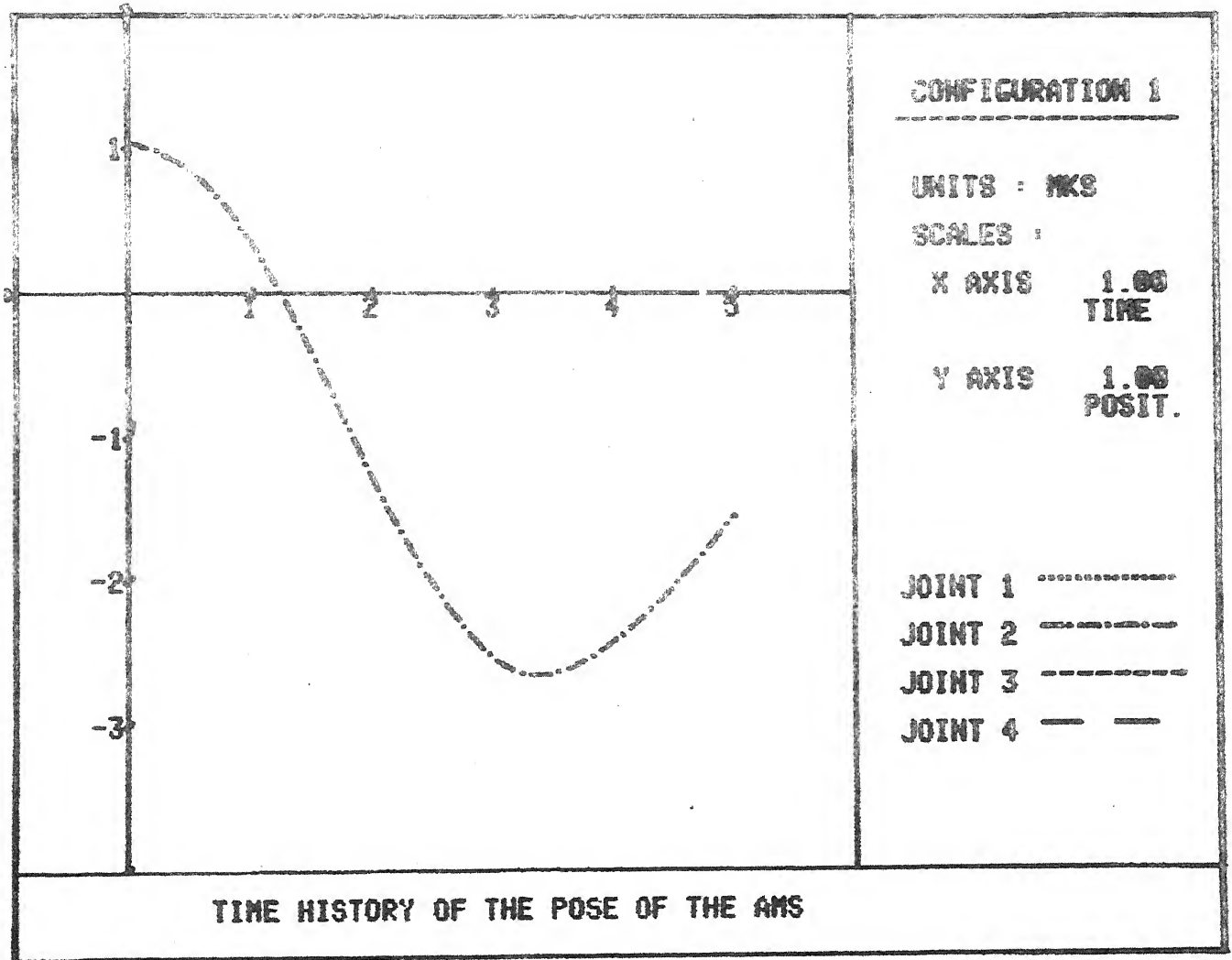












### CONFIGURATION 1

UNITS : MKS

SCALES :

X AXIS 1.00  
TIME

Y AXIS 1.00  
POSIT.

JOINT 1 .....  
JOINT 2 .....  
JOINT 3 .....  
JOINT 4 ---

